FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Design of a Body Sensor Network Embedded in Textiles for Biomedical Applications

Fardin Derogarian Miyandoab

A dissertation submitted to the Faculty of Engineering of the University of Porto in accordance with the requirements for the degree of Doctor in Telecommunications

Supervisor: Prof. João Canas Ferreira

Co-Supervisor: Prof. Vítor Grade Tavares

April 27, 2015

Design of a Body Sensor Network Embedded in Textiles for Biomedical Applications

Fardin Derogarian Miyandoab

A dissertation submitted to the Faculty of Engineering of the University of Porto in accordance with the requirements for the degree of Doctor in Telecommunications

Jury:

Prof. José Alfredo Ribeiro da Silva Matos

Prof. Maria Helena da Costa Matos Sarmento

Prof. Fernando José da Silva Velez

Prof. Rui Luís Andrade Aguiar

Prof. Rui Manuel Escadas Ramos Martins

Prof. José Alberto Peixoto da Silva

Prof. João Paulo de Castro Canas Ferreira

April 27, 2015

Abstract

Body Sensor Networks (BSNs) or Body Area Networks (BANs) refer to a subcategory of sensor networks mainly used for measuring or monitoring vital body parameters without affecting the lifestyle of human beings. These systems can be used at care centers or at home by patients or even healthy people who want to improve or monitor his/her health conditions. It is expected that in the coming years, BANs will help bring about revolutionary changes in health care systems and applications.

A BAN consists of wearable or implantable sensors or computing devices, and a communication network for data collection. Usually, an on-body system is responsible for collecting data from sensors via an intra-network and send them to a computer or PDA via a wireless link. According to the application and design parameters, the intra-network can be wired, wireless or even use the human body as communication medium.

The purpose of the work described in this dissertation is to design and evaluate a new wearable BAN to support measuring human locomotion parameters in the most practical, comfortable and non-invasive way. The entire communication infrastructure, including hardware, software and protocols, has been designed for this purpose. Although the design of the signal acquisition modules, driver software and data processing programs are beyond the scope of the present work, the designed communication system was embedded together with those facilities and tested in real data-acquisition experiments. This dissertation then focus on the networking component, in its different levels. The proposed system can also be used in many other BAN applications, especially those including a large number of sensors in BAN scale, embedded in textile and with high data-rate communication demands. The overall system includes an on-body central processing module connected to a computer via a wireless link and a wearable network sensor. Due to the fixed location of the sensors and the possibility of using conductive yarns in textiles, a wired network has been considered for the wearable components. Employing conductive yarns instead of using wireless links in the wearable unit provides a more reliable communication, higher data rates and throughput, and less power consumption. The wearable unit is composed of two types of circuits including sensor nodes and a base station, all connected to each other with conductive yarns forming a mesh topology with the base node at the center. In comparison with traditional serial or star topologies, a mesh connection is more reliable, provides higher data-rate communication and supports significantly more sensors. From the standpoint of the network, each sensor node is a four port router capable of handing over packets from destination nodes to the base station. For experimental evaluation, all nodes were connected to existing data acquisition boards capable of acquiring electromyography (EMG) signals (electrical signals from the skin) and kinetic information (using accelerometers and gyroscopes).

Two prototypes of the sensor nodes have been designed. The first prototype is based on

packet switching. The Source Routing for Minimum Cost Forwarding (SRMCF) routing protocol has been designed to establish a spanning tree of all minimum cost forwarding paths with the base node at the center. The physical and MAC layers have been implemented on a low-power FPGA (Actel IGLOO AGLN125). The second prototype was designed to improve the performance of the system based on observed limitations when using packet switching in the first prototype. The end-to-end communication in the second prototype uses a hybrid circuit and packet switching; packet delivery from sensor nodes to the base station uses hybrid switching, while in the reverse direction, or between sensor nodes, packet switching is used. This hybrid switching scheme significantly improves system performance in terms of end-to-end delay, throughput and power consumption. The communication module of the second prototype has been implemented on an integrated circuit (IC) using a 4-metal, 0.35 µm CMOS technology. The maximum data rate of the system is 35 Mbps while supporting tens of sensors, which is much more than current BAN applications need. The IC also implements a highly precise, sub-microsecond one-way time synchronization protocol, which is used for timestamping the acquired data. The suitability of the proposed system for utilization in real applications has been demonstrated experimentally with a test setup for collecting data from movements of the lower limbs.

The main contributions of the present dissertation are:

- The SRMCF reactive routing protocol for both wired and wireless networks. The protocol has been implemented on the designed wearable system and also on TelosB wireless motes.
- Hardware prototypes of sensor nodes (excluding acquisition circuitry), base station and central processing module with physical and MAC layers implemented on FPGA. The prototypes are multi-task devices that support asynchronous communication and concurrent send/receive operations.
- Hardware implementation of a hybrid circuit and packet switching mechanism for end-to-end communication.
- A low-overhead, time-division multiplexing MAC protocol for the circuit switching mechanism implemented in the proposed BAN.
- A very small circuit implementing a fully-digital clock and data recovery mechanism.
- A one-way, sub-microsecond, highly precise time synchronization protocol and its circuit implementation.
- A low-power IC for use in the second hardware prototype of sensor nodes and base station. The IC implements a transmitter, a receiver, the network layers up to the routing layer, an SPI port, support for hybrid switching, a 2 kB RAM, and is able to perform time synchronization and communicate at data rates up to 35 Mbps.

All designs have been analyzed theoretically and validated experimentally.

Resumo

As designações Body Sensor Networks (BSN) ou Body Area Networks (BAN) referem-se a uma subcategoria de redes de sensores que é usada principalmente para a medição ou monitorização de parâmetros vitais do corpo sem afetar o estilo de vida do ser humano. Estes sistemas podem ser usados em hospitais ou em casa pelos pacientes, ou mesmo por pessoas saudáveis que querem melhorar ou controlar as suas condições de saúde. Espera-se que, nos próximos anos, as BANs tragam mudanças revolucionárias nos sistemas de saúde e aplicações.

Uma rede BAN consiste em alguns sensores ou elementos de processamento vestíveis ou implantáveis e numa rede para recolha de dados. Normalmente, um sistema vestível é responsável por recolher dados dos sensores através de uma rede interna e enviá-los para um computador ou PDA através de uma ligação sem-fios. De acordo com os parâmetros da aplicação e de projeto, a rede interna pode ser construída com fios, sem fios ou mesmo utilizando o corpo humano como meio de comunicação.

O objetivo deste trabalho de dissertação é a conceção e avaliação de um novo sistema vestível para medir parâmetros de locomoção humana, da maneira mais prática, confortável e não-invasiva. Toda a infraestrutura de comunicação, incluindo *software*, *hardware* e protocolos, pois foi projetada para este fim. Embora quer o projeto dos módulos de aquisição de sinal e respetivos controladores quer o desenvlvimento dos programas de tratamento de dados estejam fora do âmbito deste trabalho, o sistema de comunicações desenvolvido foi integrado com estes componentes e testado em experiências de aquisição de dados reais. Esta dissertação foca-se na componente de rede, nos seus diferentes níveis. Os resultados alcançados e o sistema projetado também podem ser usados em muitas aplicações BAN, especialmente aplicações que incluem um grande número de sensores embebidos em têxteis e para comunicações que requerem elevada taxa de transferência de dados. O sistema inclui um módulo de processamento central ligado a um computador através de uma ligação sem fios e uma rede de sensores vestível. Devido à localização fixa dos sensores e possibilidade de utilização de fios condutores têxteis, foi considerada uma rede com fios para interligar os sensores. Empregando fios condutores em vez de usar as ligações sem fios possibilita, uma maior taxa de transferência de dados, maior rendimento e menor consumo de energia. A parte vestível é composta por dois tipos de circuitos, nós sensores (SNS) e um estação-base (BS), todos interligados com os fios condutores que formam uma topologia em malha com o BS no centro. Em comparação com as topologias tradicionais, série ou estrela, uma ligação em malha é mais confiável, fornece comunicação com maior taxa de transferência de dados e suporta significativamente mais sensores. Do ponto de vista da rede, cada nó tem 4 portos de comunicação e atua como um encaminhador de pacotes para a estaao-base. Todos os nós sensores foram equipados com um circuito de aquisição de sinais electromiográficos (EMG) e também

com um acelerómetro e giroscópio para medir os parâmetros cinéticos.

Foram desenvolvidos dois protótipos dos nós sensores. O primeiro protótipo é baseado em comutação de pacotes. O protocolo de roteamento SRMCF foi projetado para estabelecer *spanning trees* de todos os caminhos com um custo de encaminhamento mínimo para a estação-base. As camadas físicas e MAC foram implementadas numa FPGA de baixo consumo (Actel IGLOO AGLN125). O segundo protótipo melhora o desempenho do sistema eliminando as desvantagens, observadas no primeiro protótipo, associadas à utilização de comutação de pacotes. Neste protótipo a comunicação extremo-a-extremo é estabelecida com base em comutação híbrida combinando comutação de pacotes e de circuitos: a transmissão de pacotes na direção SNs para BS emprega comutação híbrida; na direção reversa ou entre nós usa comutação de pacotes. Esta comutação híbrida melhora significativamente o desempenho do sistema em termos de atraso de extremo-a-extremo, débito e consumo de energia. O módulo de comunicação do segundo protótipo foi implementada num circuito integrado fabricado em tecnologia CMOS de 0,35 µm com quatro níveis de metal. A taxa máxima de transmissão de dados do sistema é de 35 Mbps com dezenas de sensores, o que é muito maior do que as necessidades das aplicações Body Area Network (BAN) atuais. O Application-specific Integrated Circuit (ASIC) inclui também um protocolo de sincronização sub-microssegundo unidirecional de alta precisão, que permite etiquetar os dados adquiridos com informação temporal precisa. A capacidade do sistema proposto ser usado em aplicações reais foi provada experimentalmente na recolha de dados dos membros inferiores em movimento.

As principais contribuições e novidades da dissertação são:

- O protocolo de roteamento SRMCF reativo para ambas as redes com e sem fio. O protocolo foi implementado num sistema vestível projetado e também sobre as motes sem fio TelosB.
- Protótipos em *hardware* dos nós, da estação-base e do módulo de processamento central com as camadas física e MAC implementadas em FPGA. Os protótipos são dispositivos multi-tarefa capazes de comunicação assíncrona e operações de receção/transmissão concorrentes.
- Implementação em *hardware* de um mecanismo híbrido comunicação por comutação de pacotes e comutação de circuitos.
- Um protocolo de baixo custo para multiplexagem no tempo ao nível MAC implementado para a rede BAN proposta.
- Um circuito digital muito pequeno para recuperação de relógio e dados (CDR).
- Um protocolo de sincronização unidirecional de elevada precisão (sub microsegundo) e a respetiva implementação em *hardware*.
- Um circuito integrado de aplicação específica (ASIC) para uso no segundo protótipo dos nós sensores e da estação-base. O ASIC inclui recetor, transmissor, todos os níveis de rede até ao nível de roteamento, um porto SPI, suporte para comutação híbrida de pacotes e circuitos, e uma memória RAM de 2 kB, sendo capaz de sincronização temporal precisa e comunicação a taxas de dados até 35 Mbps.

Todos os projetos desenvolvidos foram analisados teoricamente e testados experimentalmente.

Acknowledgements

I wish to thank all the people who have helped me throughout the years that I needed to complete the dissertation.

First of all I would like to thank my supervisor Prof. João Canas Ferreira. This dissertation would not have been possible without the support of his. I appreciate his guidance and right suggestions at the right time.

I would like to thank my co-supervisor Prof. Vítor Grade Tavares for his great comments, the thesis has greatly benefited from them.

I also would like to thank Prof. José Machado da Silva for his support and valuable suggestions.

A huge thanks goes to my family for their support and encourage. This dissertation is dedicated to my family, especially to my parents.

I would like to thank my friend Reza who has suggested me MAP-Tele telecommunication doctoral program.

I would like to thank Ruben for his remarkable effort to use the designed system for data acquisition.

I would like to thank my colleagues and friends, Nassim, Porkodi, Simona, Mohammad, Zhaleh, Cristina, Ganga, João, and Bruno.

Finally I would like to thank Portuguese funding organization FCT, INESC and FEUP.

Fardin Derogarian Miyandoab

Contents

List of Figures

List of Tables

Abbreviations

1-Wire	Single-Wire Serial Interface
3G	Third Generation of Mobile Telecommunications Technology
4G	Fourth Generation of Mobile Telecommunications Technology
AC	Alternating Current
ACC	Accelerometer
ACK	Acknowledgement
ADC	Analog to Digital
AFH	Adaptive Frequency Hopping
AGC	Automatic Gain Controller
AODV	Ad hoc On-Demand Distance Vector Routing Protocol
AP	Access Point
ARP	Address Resolution Protocol
ASIC	Application-specific Integrated Circuit
AWGN	Additive white Gaussian noise
B-MAC	Berkeley-MAC
BAN	Body Area Network
BCC	Body Channel Communication
BCU	Body Control Unit
BER	Bit Error Rate
BS	Base Station
BSN	Body Sensor Network
CAN	Controller Area Network
CAP	Contention Access Period
CCA	Clear Channel Assessment
CDF	cumulative density function
CDR	Clock and Data Recovery
CF	Conductive-Fabric
CFP	Contention Free Period
CMOS	Complementary Metal Oxide Semiconductor
CPM	Central Processing Module
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CTS	Clear to Send
DC	Direct Current
DC-PLC	Direct Current Power Line Communication
DJ	Deterministic Jitter

DLL	Delay-locked Loop
DNA	Deoxyribonucleic Acid
DSP	Digital Signal Processing
DSR	Dynamic Source Routing
ECG	Electrocardiography
EDA	Electronic Design Automation
EEG	Electroencephalography
EMG	Electromyography
EMI	Electromagnetic Interference
FIFO	First-in First-out
FPGA	Field Programmable Gate Arrays
FSK	Frequency-shift Keying
FSM	Finite State Machine
FTSP	Flooding Time Synchronization
GPRS	General Packet Radio Service
GPS	Global Positioning System
GYR	Gyroscope
H-MAC	Hybrid MAC
HBC	Human Body Communication
HCI	Human-computer Interface
HDL	Hardware Description Language
I2C	Inter-integrated Circuit
IC	Integrated Circuit
ICT	Information and Communication Technology
IDE	Integrated Development Environment
IR	Impulse Radio
ISM	Industrial, Scientific and Medical
LAN	Local Area Network
LEACH	Low Energy Adaptive Clustering Hierarchy
LPL	Low Power Listening
LQSR	Link Quality Source Routing
LSB	Least Significant bit
MAC	Media Access Control
MCF	Minimum-Cost Forwarding
MEMS	Micro-electromechanical
MHBC	Magnetic Human Body Communication
MICS	Medical Implant Communication Service
MS	Medical Server
MTU	Maximum Transmission Unit
NB	Narrow Band
NED	Network Description
NFC	Near Field Communication
NoC	Network-on-Chip
NRZ	non Return to Zero
NRZI	non Return to Zero Inverted
NTP	Network Time Protocol
ODMRP	On-Demand Multicast Routing Protocol

OS	Operation System
OSI	Open Systems Interconnection model
PAN	Personal Area Network
PCB	Printed Circuit Board
PD	Personal Device
PDA	Personal Digital Assistants
PDF	probability density function
PEGASIS	Power-efficient Gathering in Sensor Information Systems
PHR	PHY Header
PHS	Personal Health System
PHY	Physical Layer
PLC	Power Line Communication
PLL	Phase-Locked Loop
PMAC	Priority Guaranteed MAC
PPDU	Physical Layer Protocol Data Unit
ppm	parts per million
PS	Personal Service
PSDU	Physical Layer Service Data Unit
PTP	Precision Time Protocol
QoS	Quality of Service
RAM	Random Access Memory
RBS	Reference Broadcast Synchronization
RC	Resonance Compensator
RF	Radio Frequency
RFID	Radio Frequency Identification
RJ	Random Jitter
RTL	Register Transfer Language
RTS	Request to Send
RX	Receiver
RZ	Return to Zero
SAR	Specific Absorption Rate
SCK	Serial Clock
SD	Secure Digital
SDI	Serial Data Input
SDO	Serial Data Output
sEMG	Surface electromyography
SHR	Synchronization Header
SIPO	Serial in Parallel Out
SIR	Signal-to-Interference Ratio
SMS	Short Message Service
SN	Sensor Node
SNR	Signal to Noise Ratio
SoC	System on Chip
SOIC	Small Outline Integrated Circuit
SPI	Serial Peripheral Interface
SR	Source Routing
SRMCF	Source Routing for Minimum Cost Forwarding

SS	Slave Select
TCF	Transmission Confirmation Frame
TDM	Time-division Multiplexing
TDMA	Time Division Multiple Access
TF	Token Frame
TPSN	Timing-sync Protocol for Sensor Networks
TX	Transmitter
USB	Universal Serial Bus
UWB	Ultra Wide Band
VCO	Voltage Controlled Oscillator
WBAN	Wireless Body Area Network
WHS	Wearable Health System
WiFi	Wireless Fidelity
WMTS	Wireless Medical Telemetry Services
WPN	Wearable Personal Network
WSN	Wireless Sensor Network

Chapter 1

Introduction

One of the great hopes of mankind is living healthy and overcome illnesses and disabilities. However, in reality, diseases and health problems are inevitable. Besides health problems, the growth of the elderly population is a new challenge that is gradually getting more serious. On the other hand, health care expenditure is already considerable and rises yearly. The impending health crisis has motivated researchers, industrialists, and economists to look for new solutions. Apart from medical treatment methods and medications, biomedical systems and equipments play an important role in this context. The non-intrusive and ambulatory health monitoring of patient's vital signs with real time updates of medical records via a network connection may open the way for economical solutions to some of the challenges that health care systems face.

In recent years many promising technological advances in electronics, information technology and communication systems have paved the way to new concepts in health care. For example, telemedicine, eHospital, and ubiquitous health care are enabled by emerging new electronic devices and wireless broadband communication technologies. While initially becoming main-stream for portable devices such as tablets and smart phones, wireless communications is evolving towards wearable solutions and even implantable solutions are being introduced. These health care devices rely on sensor networks and Body Area Networks (BANs) [YH11, YK12].

The purpose of this dissertation is to design and evaluate a BAN for supporting the measurement of human locomotion parameters in the most practical, comfortable and non-invasive way. The work was carried out in the context of the ProLimb project [INE13], but the resulting communication system can carry over to many other BAN applications, especially those that include a large number of sensors embedded in textiles and require high data rates (compared to more common BANs). The implementation of the full communication system involved the design of an appropriate communication protocol, of dedicated hardware and of the controlling software, all of which have been designed from scratch for this application. Since the work includes aspects from the electronics, communication and networking fields, this chapter provides a brief general introduction to BANs and wearable systems before discussing the motivation and presenting the main research contributions of this work.

1.1 Body Area Networks

BANs are a category of sensor networks that connect wearable or implantable sensors and computing devices. A BAN consists of a set of mobile or fixed and compact intercommunicating sensors, which monitor vital body parameters and movements for medical or non-medical applications [XZ09, YH11, YK12, UHB+12b, SKLF13, DP14]. Each node has enough capability to process and forward information to a base station. In health care, a BAN can be used as the infrastructure for smart and affordable health care systems that can provide long term health monitoring of patients under natural physiological states without constraining their normal activities. They can be useful as part of diagnostic procedures, in maintenance of chronic conditions, in supervised recovery from a surgical procedure, and in the detection of emergency events [JMOG05]. The IEEE 802.15 Standard Committee formed a study group (IEEE 802.15.6 SGBAN) to specify a standard for Wireless Body Area Networks (WBANs) [XZ09, KUU10, JM11]. The ability to deploy sensor nodes on the human body creates the opportunity for developing a large number of applications not only in medical health care, but also in several other fields, such as in sports and entertainment, or military and defense.

In sensor-based applications, BANs for both medical and non-medical applications consist of three types of components:

1. **In-body and on-body sensors:** These include implantable, swallowable and wearable sensor nodes with appropriate sensors according to the phenomena to be monitored , e.g., Electromyography (EMG) and Electrocardiography (ECG).

2. **Data collection infrastructure:** This is a network of sensors connected by radio, the human body or wires, both normal copper wires or conductive yarns. In many cases, a central node is responsible for collecting the acquired signals and sending them via a wireless link to other devices (like personal computers).

3. **Personal server:** The collected data is transmitted to a computer, PDA or mobile phone via a wireless link such as ZigBee, Bluetooth or WiFi. The received data can then be saved, processed or transferred to a hospital or clinic via the Internet.

The communication in Body Sensor Networks (BSNs) is usually based on sensor and ad hoc network concepts and protocols. However, BANs have their own specific characteristics that may not be supported by or be compatible with current protocols. The main differences between a Wireless Sensor Network (WSN) and a BAN are [MW10, BBI+11, YK12, RFR+14]:

1. **Data rate and quality of service:** Depending on the application and on the type of data, the data rate requirements vary in a very broad range; e.g 1 kbps for a temperature sensor to 10 Mbps for video streaming. On the other hand, a high level of Quality of Service (QoS) should be guaranteed in medical applications. Appropriate error correction and interference-avoidance methods should be implemented in the Media Access Control (MAC) and physical layers to reduce Bit Error Rate (BER) and increase reliability.

2. **Range and topology:** In most applications, the communication range should not be larger than a few meters (3 m to 6 m). Thus, in many cases a simple star topology is usually enough; however, in the presence of obstacles to radio propagation, a multi-hop communication with a relaying technique must be established.

3. **Security:** This aspect is of primary importance, especially for medical and military applications, and it should be addressed in terms of privacy, confidentiality, authorization, and integrity.

4. **Antenna and radio channel:** Antenna design can be a critical issue because the proper trade-off between the antenna size and its efficiency needs to be considered. Moreover, the presence of the human body cannot be neglected, since it affects the antenna's radiation and polarization characteristics, according to the specific position of the device on the body. A good radio channel characterization is then mandatory in order to design an antenna that is able to provide the proper radiation properties.

5. **Power consumption:** The devices used typically have limited energy resources available. Furthermore, for many devices, it is impossible or difficult to recharge or change the batteries, although a long lifetime of the device is wanted (up to several years or even decades for implanted devices). Hence, the energy demands and consequently the computational power of such devices should be limited.

6. **Coexistence:** Most of the BANs are designed to operate in the license-free ISM band centered at GHz. This is an overcrowded radio band, since Wi-Fi (IEEE 802.11), Bluetooth (IEEE 802.15.1), IEEE 802.15.4/ZigBee and other standards operate in this band. Many WBAN applications (e.g., medical applications) require very high reliability, especially when emergency or alarm traffic has to be established; therefore, techniques to avoid or reduce interference should be studied and implemented.

7. **Form Factor:** Size constraints in BANs can be stringent; the most critical aspect in this regard is to fit the antenna and the battery into a very tight case while ensuring good radiation properties and device lifetime. This is true mainly for implantable devices; however, when a WBAN node is designed to be worn, flexibility and stretchability may be more relevant in order to be comfortable for the user, especially in sports, fitness, and military applications.

8. **Signal processing:** WBAN applications are power-limited and the radio circuits are often the most power-consuming modules of the system. However, power-efficient signal processing techniques can help the designer keep under control the power consumption related to the acquisition and analysis of the biological signals.

9. **Safety for the human body:** At the frequencies of interest for WBANs, the known health-related effects include only heating of human tissues. The International Commission on Non-Ionizing Radiation Protection (ICNIRP) specifies general restrictions and limits that

have to be met to guarantee health safety when the body is exposed to time-varying electromagnetic fields. For the frequency range of 100 kHz to 10 GHz such restrictions are defined in terms of Specific Absorption Rate (SAR), which represents the mass-normalized rate at which Radio Frequency (RF) power is coupled to biological tissues and is typically expressed in units of Watts per kilogram [W/Kg]. Low power devices, such as WBAN nodes, do not radiate enough power for whole-body SAR to be a concern, but attention has to be paid to localized SAR, which is the SAR measured in the parts of the body most exposed to RF fields.

The importance of the aforementioned characteristics of BAN depends on the type of network, wired or wireless, under consideration. A wired network, either with conductive yarns or copper wire, may have better performance in terms of data rate, QoS, security, power consumption, signal processing and safety for the human body. On the other hand, issues related to antennas, coexistence, interference and coverage range are not applicable to wired networks. It can be concluded that if a given BAN application can use both wired or wireless networks, then selecting a wired network will be advantageous. This is the alternative that was selected for the wearable sensor node platform described in this dissertation.

1.2 Wearable Health Systems

Wearable Health Systems (WHSs) are a specific category of Personal Health Systems (PHSs) [PB10b, LG06a]. They have drawn a lot of attention from the research community and industry during the last decade, as evidenced by the numerous and yearly increasing research and development efforts [Tro05, GI07, LD07, TEHO11, PPB+12, TTKS14]. These systems can be characterized as integrated platforms that are worn on the body. Typical examples include wrist-worn devices and biomedical clothes. WHSs are a solution for continuous health monitoring though non-invasive biomedical, biochemical and physical measurements. As the benefits of these systems become obvious and the attitude of users towards the application of the new technologies in health care becomes more positive, it is expected that WHSs will expand significantly in the near future. For these reasons, the WHS field has been attracting increased interest from various technological fields. In recent years, different portable devices for kinematics monitoring of human movement using Micro Electromechanical (MEMS) integrated circuits with miniaturized accelerometers and gyroscopes have been introduced. These have been successfully applied in areas that go from entertainment to sports and rehabilitation, in both research and clinical environments, as well as in daily and leisure activities [JMOG05, TTT+09].

Surface electromyography (sEMG) and kinetics monitoring equipments are less frequently used in gait analysis, although recent developments have been made, including also wireless capabilities and improved wearability features. Examples include a wearable EMG-based human-computer interface (HCI) for wheelchair users with severe disabilities [MLCM05], and a cable-free network for wearable computing systems [ASS+08]. Nevertheless, the use of these systems

is still very much in the research field and very few systems, if any, are appropriate for clinical or routine utilization. The main issues or drawbacks are twofold: the vast amount of wires required just to accede a subset of the limb muscles demands new technological developments (e.g., in electronic textiles and EMG matrices); and the complex data calls for advanced signal processing and analysis tools with intuitive and simpler user interfaces.

Developing products capable of monitoring the different quantities that are involved in human locomotion, such as linear and angular movement of thighs, shanks and feet, together with the myoelectric signals of the surface muscles on the limbs requires resorting to more advanced technologies to eliminate lose interconnecting wires. There are several designs of wireless-enabled garments, with embedded textile sensors, for simultaneous acquisition and continuous monitoring of biomedical signals like ECG, respiration, EMG and daily activity. This approach has lead to the concept of smart cloth which embeds strain sensors based on piezoresistive yarns, and fabric electrodes realized with metal-based yarns [LG06a]. However, most of these developments are yet in a prototype stage, with open issues including techniques for on-body sensing, context awareness, user friendliness, power autonomy, intelligent data processing and interaction with professional medical services.

The use of fabrics with embedded conductors can provide the connections between the nodes and would make the product more user-friendly and comfortable. Previous developments show that the most comfortable and easiest way to monitor physiologic signals consists in using garments [GI07, LD07]. Among the existing available approaches, the mesh topology is the one that offers more advantages. In a mesh network, devices are connected with many redundant interconnections between network nodes such as routers and switches, and if any link or node fails, there are many other ways for two nodes to communicate. In the next section, the motivation for designing a new wearable garment with a mesh topology is presented.

1.3 Research Motivation

One of the characteristics of today's BANs is the limited number of nodes they present. Currently, the use of wireless sensors or conventional star or serial bus topologies meets the needs of many applications with few nodes. However, due to the growing use of these devices in many medical and non-medical fields, and the increase in the number of nodes and components that need to communicate with each other, the aforementioned topologies may not satisfy the requirements of the new applications envisaged. As mentioned earlier, the work for this dissertation was conducted in the context of the ProLimb project, which, compared to other BAN networks, requires a relatively large number of sensors embedded in textiles. Hence, in this dissertation a BAN based on a mesh topology is proposed, which supports applications that require a large number of sensors. Due to fixed location of the sensors on the body and the possibility of using conductive yarns in textiles, the embedded sensors are connected to each other with conductive yarns. Such a topology, together with appropriate protocols and communication circuits, leads to a high data-rate,

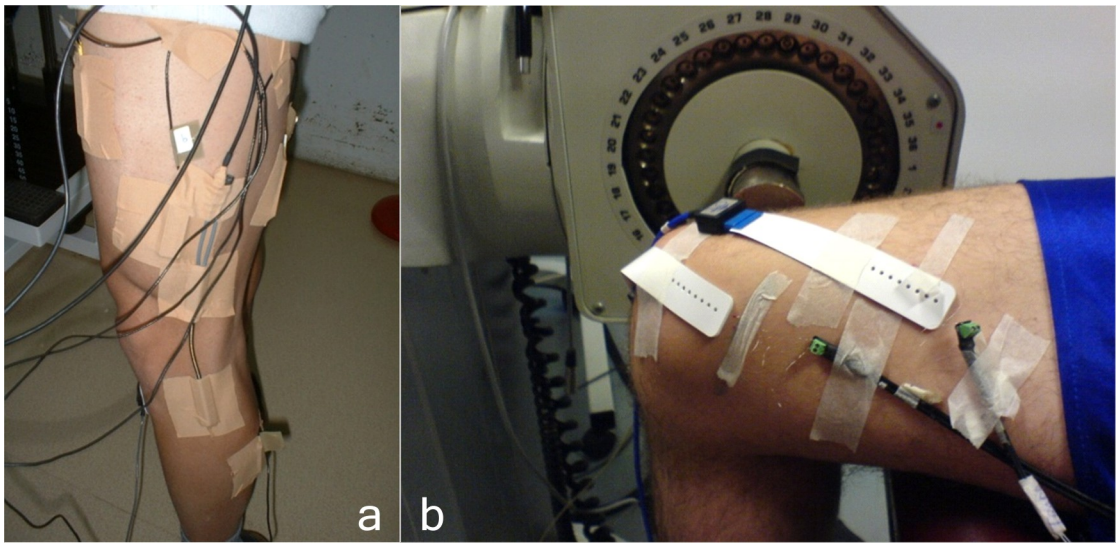


Figure 1.1: a) Lab setup for EMG assessment in four leg muscles plus one goniometer, b) Experimental setup with EMG arrays applied on the thigh, [from ProLimb project proposal]

high-throughput, low-power and reliable system. The proposed solution optimizes system performance in terms of each of the aforementioned parameters referred in Section 1.1. Although the whole system has been designed from scratch, the research described here focused mainly on the wearable platform.

1.3.1 The ProLimb Project

ProLimb Project is aimed to design a system to measure human locomotion parameters in the most practical and non-invasive way possible, even for those persons that have strong impairments or disabilities. This section briefly introduces the ProLimb project.

Locomotion is a fundamental function in the active life of a human individual at the personal, professional and social levels. Unfortunately, several neuromuscular or muscle skeletal disorders can significantly impair human locomotion or produce gait abnormalities. Among them are cerebral palsy, Parkinson disease, and hemiplegic or diabetic neuropathy, just to name a few. The analysis of normal or pathological gait is applied in the diagnosis of these disorders and in the assessment of all types of treatment programs. Current instruments and methods for gait analysis are still expensive and complex, difficult to apply by health care staff, difficult to use and uncomfortable for the patient (Fig. 1.1). Additionally, they also require a very high level of expertise for data gathering, analysis and interpretation.

The aim of the ProLimb project was to to develop an approach that not only includes the measurement of typical kinematic variables of the lower limbs (i.e., linear and angular movement of thighs, shanks and feet), but also the acquisition of myoelectric signals of the surface muscles that are most important for locomotion. The major potential of this multi-factor movement analysis is to shed light on the role of different factors in a given pathology. Such a procedure can help

clinical staff make decisions about the therapy. A key requirement for the measurement infrastructure is to be safe for the patient. The system is wearable and autonomous to allow monitoring of the relevant variables, not only in the clinical or laboratory environment, but also during daily activities of the user, with minimal interference and discomfort.

This type of measurement system will be useful in three main application scenarios:

1. To provide objective data in evaluation of gait disorders. This is very important from a clinical stand point of view, since the interaction between mechanical and neural factors is determinant and cannot be directly estimated by visual inspection or manual analysis. It will also be possible to analyze the modifications caused by therapeutic interventions.

2. If the instrument is simple and comfortable enough for the user, he/she can use it for progress self-assessment of physiatrist treatment. It can also be used as a biofeedback system to improve performance in locomotion and avoid falls, the major cause of hip fracture in the elderly.

3. It can be used for movement analysis and performance studies in sports or in physically demanding occupational activities.

In this dissertation two prototypes of a wearable sensor platform are discussed; their suitability for use in ProLimb project has been confirmed experimentally by acquiring data from the lower limbs in locomotion.

1.3.2 Research Goals

According to the research motivation and the objectives of the ProLimb project, the main goals of the dissertation work are as follow:

1. Definition of the application requirements;

2. Design of a BAN for a wearable platform with embedded sensor nodes connected in a mesh topology with conductive yarns;

3. Design of a Central Processing Module (CPM) module to make a wireless connection between the wearable modules and a computer or Personal Digital Assistant (PDA) for real-time transfer of the acquired signals;

4. Design of the essential protocols for achieving energy-efficiency, high throughput, high data rate (> 10 Mbps), high number of the sensors (>50) and reliability;

5. Design of the Sensor Node (SN) (communication and networking component) and Base Station (BS) prototypes, their analytic and experimental evaluation and their use for collecting data in a real application;

6. Design of highly precise time synchronization method for time stamping and Time-division Multiplexing (TDM) communication;

7. Use the results of the first prototype to design an IC-based second prototype with improved system performance.

The pursuit of the above goals produced further results as explained in the next section.

1.4 Research Contributions

To achieve the research goals, a BAN system has been designed which includes a wearable platform and an on-body CPM for the connection with a computer or PDA (via Universal Serial Bus (USB) port or wireless Bluetooth link). The wearable platform is composed of two types of nodes (SNs and BS) and also of their interconnections in a mesh topology, all embedded in a textile. Each node has four bidirectional communication ports and SNs act as routers to handover packets from destination nodes to the BS, which is responsible for collecting all data and sending them to the CPM. The SNs and BS (which also acts as as a sensor node) have been equipped with electromyography (EMG) electrodes for capturing electrical signals from skin tissue, and accelerometers and gyroscopes for measuring kinetic parameters.

Two prototypes of SNs and BS have been designed. The first prototype is based on packet switching; each intermediate SN stores received packets and then forwards them to the next node on the path to the BS. All the paths between the SNs and BS are predefined minimum-cost paths managed by the Source Routing for Minimum Cost Forwarding (SRMCF) routing protocol developed for this purpose. The physical and MAC layers have been implemented on Field Programmable Gate Arrays (FPGA) and the upper layers on a microcontroller.

The second prototype was designed to improve the system performance based on the observed limitations when packet switching is used in a contention-based network in the first prototype. The end-to-end communication in this one is based on hybrid circuit and packet switching: packet delivery in the direction from SNs to BS uses circuit switching and/or packet switching; in the reverse direction or from SN to SN only packet switching is used. Such a hybrid switching approach significantly improves the system performance in terms of end-to-end delay, throughput and power consumption. The communication module for the second prototype has been implemented on an Integrated Circuit (IC) fabricated in 0.35 µm CMOS technology. The IC also includes a highly precise, one-way time synchronization protocol, which is used for timestamping of the acquired data. The novel contributions of this research can be summarized as follows:

- **Routing Protocol:** For multi-hop, end-to-end packet delivery, a routing protocol is needed. Although many routing protocols for both wired and wireless networks have been designed, there is still no standard protocol for wearable networks. In this research, reactive, energy-efficient routing protocol called SRMCF has been developed, analyzed and implemented. This protocol combines Source Routing (SR) concepts (from ad hoc networks) and Minimum Cost Forwarding (MCF) methods (from heterogeneous WSNs) and can be used with both wired wearable sensor networks and WSNs. As a reactive protocol, SNs maintains no information about the network topology, but packets (from sensors to BS and vice-versa)

always communicate over paths with minimum cost. The performance of the protocol has been evaluated analytically and experimentally and compared with the MCF protocol on both wired and wireless networks.

- **Hybrid circuit and packet switching:** Power consumption, throughput and end-to-end delay are some of the parameters adversely affected in wearable systems when solely packet switching is used. The use of circuit switching in the MAC layer is one of the methods that can overcome the aforementioned drawbacks. A bufferless method is introduced which is based on circuit switching for many-to-one (SN to BS) packet delivery. The SRMCF protocol determines all paths form SN to BS with minimum cost forwarding. These paths can be used to establish circuit paths between a node and the BS in the MAC layer. So, circuit switching is based on cross-layer information exchange. As a result, communication uses a combination of both circuit and packet switching. The communication from SN to BS can also be established with a combination of both circuit and packet switching. This means that intermediate nodes are able to operate as an end node in circuit switching, receive the packet and send it on to the BS using another circuit switching or packet switching. Experimental evaluation and simulation results on the hybrid mode operation show better performance of the network in comparison with the first approach based solely on packet switching.

- **A TDM MAC protocol over circuit switching:** In a network based on circuit switching, random generation and transmission of packets leads to significant degradation of the throughput. This happens because of a high request failure in the shared nodes and links. To avoid that behavior and ensure higher likelihood of path creation and packet delivery, a time scheduling method has been introduced in the form of a TDM MAC protocol. In this way, the operation time of the system is divided in slots (with the appropriate duration) for packet carrying. Results show that such a scheduling method allows each node to use the shared environment for sending packets without collision and significantly increases the throughput of the system.

- **Clock and data recovery mechanism and circuit:** Synchronization is an important feature of many networks and distributed systems. A fully-digital, open-loop and low-complexity (small size) fast-lock synchronization circuit for Clock and Data Recovery (CDR) has been developed for this work. Synchronization is based on the open-loop selection of the correct phase of the receiver clock synchronously with the incoming signal. The clock generator of the receiver is an autonomous oscillator set to operate at the same nominal frequency. The circuit lock time is at most one clock cycle, which is faster than all methods based on PLLs or DLLs. The fully-digital circuit (including NRZI decoder) consists of only 8 logic elements in total and occupies 0.0022 mm^2, 0.35 μm CMOS process, a smaller implementation than many existing circuits. The circuit is suitable for all systems that, like the platform developed in this work, tolerate some jitter, but require fast lock time, small size and low energy consumption.

- **Time synchronization protocol and circuit:** A one-way master-to-slave, highly precise time synchronization protocol and its implementation in hardware level as a fully digital circuit has also been developed. The circuit is designed to perform synchronization in the MAC layer, so that the deterministic part of the clock skew between nodes is kept constant and compensated with a single message exchange. In each SN, the synchronization circuit provides a programmable clock signal and a real-time counter for time stamping. Experimental results show that the circuit keeps the one-hop average clock skew below 4.6 ns and that the skew grows linearly as the hop distance to the reference node increases. The sub-microsecond average clock skew achieved by the proposed solution satisfies the requirements of many wearable sensor network applications.

- **Integrated circuit for communications:** The protocol layers of the second prototype up to the network layer have been implemented in an Application-specific Integrated Circuit (ASIC) fabricated in a 0.35 µm 4-metal CMOS technology with 2.42 mm×2.42 mm in total size. The ASIC is equipped with 4 bidirectional ports, all supporting baseband communication with Non Return to Zero Inverted (NRZI) line coding and data rates up to 35 Mbps. The circuit supports circuit, packet and also hybrid switching modes. It includes a 2 kB RAM that is used to buffer the packets. This buffer and also all the registers can be read and written by a microcontroller or microprocessor via a Serial Peripheral Interface (SPI) port. The ASIC also includes all the circuits for implementing time synchronization in MAC layer. The time stamp value is readable as a register, and a programmable, globally-synchronized clock signal is made available on a pin for use in the node. The ASIC drives the conductive yarns directly and can be used to make a mesh network of the sensors in any wearable systems. The ASIC was used in the second platform prototype and confirmed to operate as expected.

1.5 Dissertation Outline

The rest of this dissertation is organized as follows.

- **Chapter 2**: This chapter presents the background and state-of-the-art related this dissertation, including BAN, WSN, wearable systems, network types and layers, nodes and circuits, CDR techniques and timestamping in sensor networks.

- **Chapter 3**: The SRMCF routing protocol is presented and analyzed in this chapter. This protocol is a reactive, energy-efficient routing protocol, which can be used in both wearable wired and wireless sensor networks. The implementation of the protocol in C and C++ using the OMNeT++ (component-based simulation software) and Contiki (operating system for the internet of things) is presented. Then the simulation and experimental results are presented and discussed.

- **Chapter 4**: This chapter presents the first prototype of the wearable platform and all the hardware developed, including both the wearable network and the wireless link. The hardware platform consists of SNs, BS, CPM and the connections between the SNs using conductive yarns. The behavior of the sensor network and the characteristics of the system are validated. Experimental and simulation results (together with examples of acquired signals) are presented.

- **Chapter 5**: This chapter presents and evaluates the IC features related to node synchronization. First, an open-loop, low-complexity (small size) fast-lock CDR circuit is presented. Then, a highly precise time synchronization protocol suitable for hardware implementation is described. For both cases, results obtained both from simulations and from implemented circuits are discussed and evaluated.

- **Chapter 6**: This chapter presents the second prototype of the hardware infrastructure, which is based on an IC implementation of the communications module, featuring a TDM MAC protocol that exploits cross-layer information, hybrid circuit and packet switching. This prototype presents significant improvements in terms of power consumption, number of buffering operations in intermediate nodes, and end-to-end delay. The IC also features high-precision time synchronization implemented in the MAC layer. The functionality of each module of the communication IC is described and the experimental results are presented. This chapter also includes a performance evaluation of the protocol based on simulation results.

- **Chapter 7**: The final chapter summarizes the outcomes of the work (including publications) and describes some directions for future research in this area.

At the end of dissertation, three appendixes present extra technical information about the first and second prototypes, testbenches, and also more captured results obtained with the system.

Chapter 2

Background and State of the Art

As mentioned in the previous chapter, a BAN consists of several components set in mutual collaboration to enable body activity monitoring. In this chapter, the background and current state of the BAN components relevant to this dissertation are presented. First, a short summary of the initial work on BANs is provided. The following section presents the communication methods used in BANs and their advantages and disadvantages are listed and compared. Next, the network layers of the system are explained and the related reported work, concerning each layer, is presented. Clock and data recovery, and time synchronization are explained in the two following sections. Then, some sensors utilized in BANs are described. The last section presents the work related to wearable BAN platforms.

2.1 Initial Work on Body Area Networks

The development of Personal Area Networks (PANs) grew out of the research done by a number of different groups working at MIT (Massachusetts Institute of Technology) in the 1990's [RRR13]. Their idea was to interconnect information appliances that were carried on the body. They could send data through the body by modulating electric fields and determine relative positioning using electric field sensing. At that time, Thomas G. Zimmerman developed technology that had the effect of allowing the body to act like a copper cable [T.G96]. The Physics and Media Group in MIT led by Neil Gershenfeld was applying a method known as "near-field coupling" to particular problems [GRA+95a]. This method allows an accurate determination of the position of one part of the body in relation to another. They placed pairs of antennas on parts of the body (e.g., on the hand and elbow), and it was noted that by running an electric current between them, the capacitance of the circuit was changed as the parts moved (e.g, by flexing the elbow). So, the determination of the positions of the antenna was possible by measuring the capacitance change. However, such a measurement method returns wrong results when a hand is placed between the antennas. Zimmerman solved the problem and showed that some of the electric current was passing through the body, therefore affecting the measurement [GRA+95b].

At the same time, a group working at the Media Lab asked these last two researchers to develop a network to connect together all the electric gadgets that a person might be carrying on them. Many people carried around a number of digital devices such as a mobile phone, a PDA and a digital watch, but none of them could communicate with each other. Zimmerman and Gershenfeld were involved in the project and noticed that if they modulated the electric field flowing through a person's body, they could make it represent a 1 or 0 symbols, thus allowing the body to act as a communication medium to carry digital information. They also observed that if the frequency and power used were both kept very low then the signal would not propagate far beyond the body. This would mean that only devices on the body or in direct contact with it were able to detect the signal. The current and voltage level used were very small, totally unnoticed by the person. The aforementioned ideas and efforts to developed a PAN were the earliest attempts at creating a BAN.

2.2 Communication Methods

The communication between on-body parts and a computer or PDA is done over a wireless link but communication among the nodes in the wearable part may be established in a different way. The wearable part of the system can be categorized according to the communication medium. In this context, the available technologies are wired, wireless and Human Body Communication (HBC). In each of these cases, a large amount of research has been done and several commercial products have been developed. Each method has its own advantages and disadvantages, which are described next.

2.2.1 Wireless Communication

Typically, a wireless sensor uses an RF transceiver for the physical layer. Most of PAN or BAN devices use wireless communication schemes such as Near Field Communication (NFC), or Body Channel Communication (BCC) [YH11]. They can be categorized in two groups: near-field inductive coupling and far-field electromagnetic communication. The inductive coupling is effective for short distance "through-body" communication connecting for example implanted devices with on-body monitoring devices. However, far-field communication based on PAN and BAN is most widely used for on-the-body sensor networking. The main advantage of utilizing wireless devices is their easy deployment, both for adding new devices or removing them. On the other hand, they have several drawbacks in this context:

1. **Energy consumption:** Despite many efforts and developments in the past years to reduce the power consumption of the wireless devices, they typically still are the most power consuming component of a sensor node.

2. **Data rates:** The data rate of Bluetooth 1.0 devices is 1 Mbps and version 3 provides up to 24 Mbps. These are the over-the-air rates, but for the application layer the values will be much smaller. IEEE 802.11 and Ultra Wide Band (UWB) provide 54 Mbps and 480 Mbps

data rate, respectively, but because of their high power consumption they are not usable in many energy-constrained applications [MAL+14].

3. **Signal propagation:** Signal attenuation, reflection, barriers and multipath effects can affect the quality of the signal transmission. Although the distance in BANs is short, the human body can shadow the signal, mainly because the water in the body absorbs 2.4 GHz waves.

4. **Interference:** Most BAN devices equipped with a wireless interface such as Bluetooth, work in the 2.4 GHz Industrial, Scientific and Medical (ISM) band. Although Bluetooth, is able to cope with 802.11 by hopping to different channels, there is a limit to this capability. In addition, it may even interfere with Radio Frequency Identification (RFID) devices [CGV+11].

5. **Security:** Wireless communication uses a shared medium,so eavesdropping is easy and encryption is necessary to ensure privacy.

6. **Health Issues:** BAN devices operate at microwave frequencies or below. The electromagnetic radiation in this band is known as non-ionized, that is it can appear as thermal energy and be harmful to the body. But it should be noted that the signal power for such a low power equipment is too small to generate any detectable effect on the body. Nevertheless, companies are taking SAR regulations into consideration [Org14, oNIRPI14]. These regulations define the maximum exposure to electromagnetic fields that a device may impose on a human body so as not to cause any health problems.

2.2.2 Wired Communication

The aforementioned issues related with wireless devices and the possibility of having available conductive yarns or copper wires lead to an alternative use of wired communication in BANs. Wired networks, as a second type of communication infrastructure for BAN applications, provide high-speed, reliable and low-power solutions [NJM04, LPN+10, MHT+05]. The use of fabrics with embedded conductors can provide the required electrical connections, and would make the product more user friendly and comfortable. Previous developments show that the most comfortable and easiest way to monitor physiologic signals consists in using garments [LG06b, GI07], with conductive yarns used as wire lines. The design of such a wearable system must take into account the fact that electrical characteristics of conductive yarns are not exactly those of normal wires, since their electrical properties may exhibit significant variations in relaxed and stretched modes. In addition, conductive yarns are prone to tearing and wearing out; thus, to achieve high reliability the network must be tolerant of catastrophic and parametric faults. Wired networks also have some drawbacks:

1. **Node Placement:** Adding, removing or replacing the nodes may not be easily done, especially if the sensors are embedded in the textile.

2. **Lack of Standards:** Protocols previously designed for other networks such Inter Integrated Circuit (I2C), Controller Area Network (CAN), Single-Wire Serial Interface (1-Wire), SPI or serial bus are applicable to wearable wire networks but there still is no standard protocol.

3. **Cumbersome:** In many applications, especially when involving mobility, copper wires are intrusive, cumbersome and aesthetically unpleasing.

2.2.3 Human Body Communication

HBC is a communication technique in which the body is used as signal propagation medium [YH11]. Electrical signals can easily propagate through human body and recently several HBC schemes have been suggested for BAN where the data is transferred through the skin of the human body [AC14, KKK14] . Because the body becomes the transmission pathway for electronic signals, in comparison with traditional wireless communications that use space propagation, human body communications are superior in information privacy and they are more energy-efficient. Furthermore, because a communication path can be established through contact with another communicating party, human body communications may be used in man-machine interfaces that utilizes the movement of people. Since communication is restricted to the human body, interception is more difficult than in other wireless methods. Another advantage is its low transmission power, which is lower than for wireless (electromagnetic) but higher than for wired communication [AC14]. The low-power operation makes it less troublesome with regards to SAR compliance. There are however issues regarding interfacing with the body (probe sizes), skin irritation, lake of communication standard and low data rates and reliability [MAL+14].

2.3 Communication Architecture of Body Area Networks

In most BAN applications, the communication architecture can be categorized in three main classes as shown in Fig. 2.1 [YK12, CGV+11]:

1. **Intra-BAN communication:** This part of the system includes in-body or on-body sensor nodes, a central node and a network between them, which can be wired, wireless ($< 2\,\mathrm{m}$ coverage) or HBC. A central node or Personal Service (PS) collects data from the sensors and send it to an external destination.

2. **Inter-BAN communication:** This communication class refers to the communication between the PS and one or more Access Points (APs). The APs can be part of a fixed infrastructure or they can be deployed dynamically to handle specific situations like disasters or other serious events. Inter-BAN communication aims to interconnect BANs with various networks which can be easily accessed in daily life (including cellular networks and the Internet). The paradigm of inter-BAN communication is divided in two subcategories, as follows [MAL+14]:

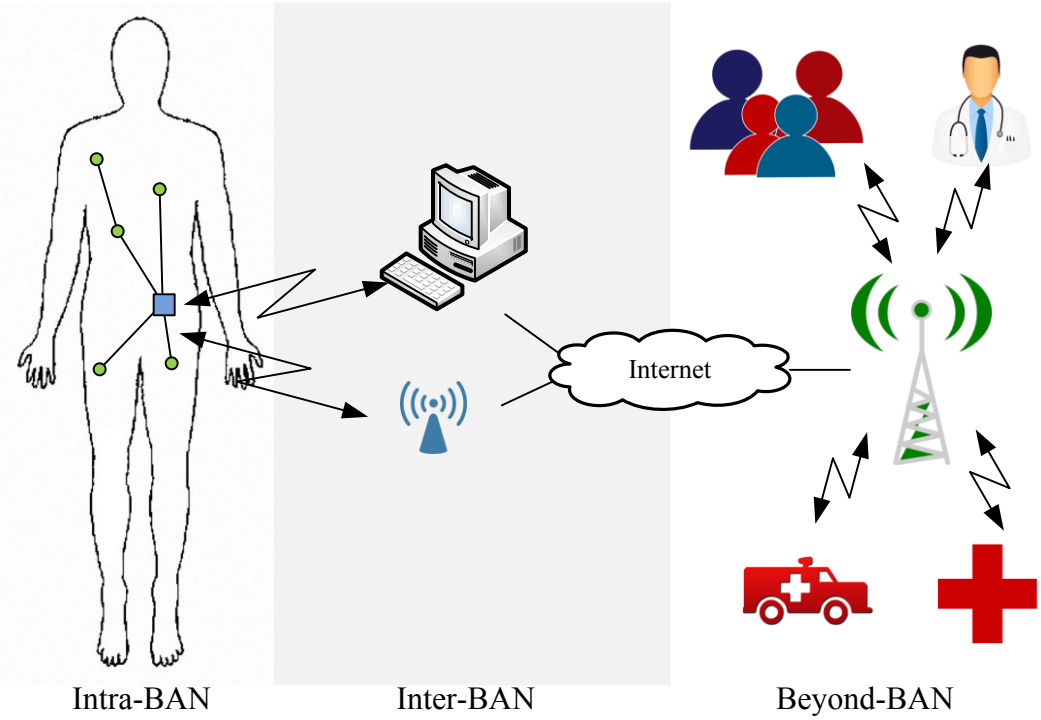


Figure 2.1: Communication architecture of BAN.

.

- **Infrastructure-based architecture:** This architecture is used in most BAN applications, as it facilitates dynamic deployment in a limited space, such as a hospital, as well as providing centralized management and security control. The AP can act as a database server related to its application.
- **Ad hoc based architecture:** In this architecture, multiple APs transmit information. The APs in this architecture usually assume a mesh configuration that enables flexible and fast deployment, allowing for the network to easily expand, provide larger radio coverage due to multi-hop dissemination and support patient mobility. This configuration has a much larger coverage range than an infrastructure-based architecture, and therefore facilitates movement around larger areas. In fact, this interconnection extends the coverage area of BANs from 2 to 100 meters, which is suitable for both short and long term setups [CGV+11].

3. **Beyond-BAN communication:** This class of communication is for use in metropolitan areas. A gateway such as a PDA is needed to bridge the connection between the inter-BAN and this network [LBM+11]. However, the design of beyond-BAN communication is application-specific. Typically, in a medical environment the database is one of the most important components of the system as it includes the medical history and profile of the user. Thus, doctors or patients can be notified of an emergency status through either the Internet or a Short Message Service (SMS). Additionally, beyond-BAN communication allows accessing (or restoring) all necessary information of a patient to help on diagnosis

and treatment [CGV+11]. In some applications, it is also possible for the the PS in the intra-BAN segment to use GPRS/3G/4G instead of talking to an AP (avoiding the inter-BAN segment) [JLH+08].

2.4 Types of BAN Nodes

Nodes of a BAN are defined as independent devices with communication capabilities. Nodes can be classified by functionality, implementation and role in the network [MAL+14]. The classification of nodes based on functionality is as follows:

1. **Personal Device (PD):** This device collects all the data received from sensors and actuators and handles interaction with other users. The PD then informs the user. In some applications, the PD may also be called body gateway, sink, Body Control Unit (BCU) or PDA [LBM+11].

2. **Sensor node:** Sensors in BANs measure certain parameters of the body (either internally or externally). They gather data, process and provide response to information. These sensors are either physiological sensors, ambient sensors or biokinetics [LBM+11, MHA+09]. Some existing types of these sensors could be used in one's wrist watch, mobile, or earphone and consequently, allow wireless monitoring of a person anywhere, anytime and with anybody. A list of different types of commercially available sensors used in BANs are as follows: EMG, Electroencephalography (EEG), ECG, temperature, humidity, blood pressure, blood glucose, pulse oximetry (SpO_2), CO_2 gas sensor, spirometer, plethysmogram, DNA sensor, magnetic biosensor, transmission plasmon biosensor, and motion (gyroscope and tri-axial accelerometer) [RCL+12, BBA14].

3. **Actuator node:** An actuator interacts with the user upon receiving data from the sensors. It is responsible to provide feedback in the network by acting on sensor data, for example pumping the correct dose of medicine into the body in health care applications [WP10].

IEEE 802.15.6 has proposed another classification for nodes in a BAN based on the way they are associated with the wearer's body [XZ09, YS09]:

1. **Implant Node:** This type of node is implanted in the human body, either immediately underneath the skin or inside the body tissue.

2. **Body Surface Node:** This type of node is either placed on the surface of the human body or 2 centimeters away from it.

3. **External Node:** This type of node is not in contact with the human body and rather a few centimeters to 5 meters away from the human body.

The classification of nodes in BANs based on their role in the network is as follows [MAL+14]:

1. **Coordinator:** The coordinator node is like a gateway to the outside world, another BAN, a trust center or an access coordinator. The coordinator of a BAN is the PDA, through which all other nodes communicate.

2. **End Nodes:** The end nodes in BANs are limited to performing their embedded application. However, they are not capable of relaying messages from other nodes.

3. **Relay:** Relay are intermediate nodes and they have a parent node, possess a child node and relay messages. In essence if a node is at an extremity (e.g. a foot), any data sent is required to be relayed by other nodes before reaching the PDA. The relay nodes may also be capable of sensing data.

2.5 Sensors

Advances in electronic and communication systems have enabled the development of low-cost, low-power, energy-efficient, multi-functional sensors [KW05]. As the electronic technology advances, it becomes possible to design miniaturized biomedical devices to place them on or inside the body, e.g., wearable devices for physiological signal detection (e.g. pressure, blood flow, ECG, EMG and EEG), pacemakers, implantable devices, and wireless endoscopes (i.e. electronic pill). Implantable drug delivery, micro robots in surgery [JKJ10], and deep brain stimulation/recording for brain-computer interfaces [MKC09] are some examples of biomedical sensors. In the future, there will be many more body sensors based on new concepts developed to improve health care by monitoring important body signals. Current research projects in the area of body sensors aim at improving the performance and reducing the power consumption for longer operation time [Yuc10].

Biomedical sensors benefit greatly from micro- and nanoelectronics. Higher accuracy and smaller form factor combined with reduced cost and increased convenience of use are enabled by incorporation of Complementary Metal Oxide Semiconductor (CMOS) IC design in the realization of biomedical systems. The compact hearing aid devices and current pacemakers are some of the examples of how CMOS ICs bring about these new functionalities and services in the medical field [YH11]. In recent years, many researchers are trying to develop new biomedical solutions such as Artificial Retina, Deep Brain Stimulation, and Wearable Healthcare Systems. These devices are available by combining the recent advances of biomedical technology with low power CMOS IC technology. Biomedical CMOS ICs have different characteristics from other CMOS ICs. These ICs will be used in very intimate contact with the body. Therefore, biological effects and interactions related to the specific application should be considered before the chip design and after chip fabrication. Figure 2.2 depicts a generic architecture of a sensor node for biomedical applications [Ser10, YH11].

Biosensors started with the development of enzyme electrodes by scientist Leland C. Clark [SE06]. Since then, research communities from various fields, such as VLSI, physics, chemistry, and material science, have come together to develop more sophisticated, reliable, and mature

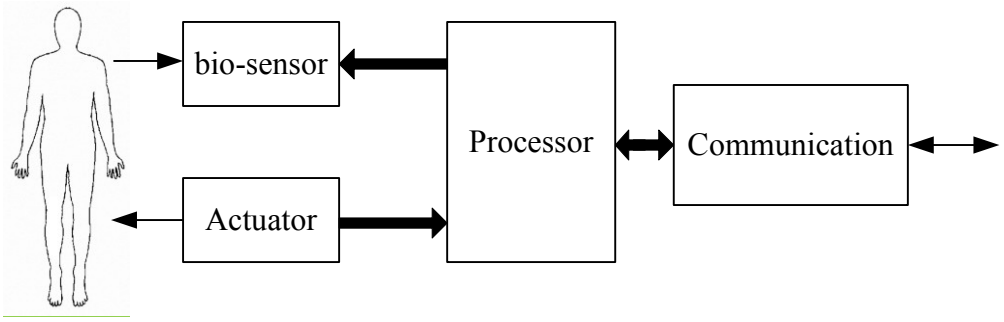


Figure 2.2: Typical architecture of a biomedical sensor.

biosensing devices. Applications for these devices are found in the fields of medicine, agriculture, biotechnology as well as the military and bioterrorism detection and prevention. Sensors convert signals of one type into an equivalent signal of another type, for example, an electrical signal [Kon05]. Biomedical sensors take signals representing biomedical variables and convert them into what is usually an electrical signal. As such, the biomedical sensor serves as the interface between a biologic and an electronic system and must function in such a way as to not adversely affect either of these systems. Biosensors can have a variety of biomedical, industry, and military applications [SE06].

2.6 Related Work in Wearable Networks

In the last years, a number of wired and wireless networks for wearable applications have been proposed. Several surveys on BAN networks and wearable system can be found in the literature of recent years [CLCC09, MW10, GVLC+13, RRR13, RFR+14, MAL+14]. As for the surveys on applications, a special emphasis on medical ones and healthcare is given in [PB08, UKU+09, PB10a, WTZS10, JM11, BSM+12].

One of the early attempts to manufacture a wearable system was reported by Post and Orth [PO97]. They built an electronic system by utilizing conductive yarns as a data bus and described some techniques for building circuits from commercially available fabrics, yarns, fasteners, and components. They also have shown that all of the input devices can be made by seamstresses or clothing factories, entirely from fabric.

Based on theoretical analysis and experimental results, the ability of conductive yarns to carry both data and power supply has been shown by Wade and Asada [WA07a]. They introduced a wearable DC-PLC power line communication network for health sensing and rehabilitation. Their system provides a network through which multiple sensors and actuators can send and receive both information and power. Their method applies to wires or conductive yarns with high conductivity.

A Wearable Personal Network (WPN) system based on a fabric serial bus with conductive yarns was introduced by Hyung et al. [LPN+10]. Their system is a four-layered wearable PAN using a bus topology controlled by a host node and their protocol works with a variety of microcontrollers. The system was implemented on an FPGA running at 50 MHz and works at up to 10 Mbps.

The profile layer is provided to make the application development process easy. The data link layer exchanges frames in a master-slave manner in either the reliable or best-effort mode. The lower part of the data link layer and the physical layer of the WPN is built on a fabric/serial-bus interface that is capable of measuring bus signal properties and adapting to medium variation. They have implemented WPN communication modules (WCMs) on small flexible printed circuit boards and they have also designed a WPN shirt prototype using implemented WCMs and conductive yarns.

An inductive transceiver with a fault-tolerant network switch for multi-layer wearable BAN applications has been introduced by Yoo and Lee [YLY09]. The inductive coupling transceiver is made from conductive yarns and employs a Resonance Compensator (RC) with a digitally controlled on-chip capacitor bank and a variable hysteresis Schmitt Trigger to compensate dynamic and static variances of the woven inductor. It communicates at a 10 Mbps data rate. Each node consists of two chips; transceiver and switches, and resonance compensator, both fabricated in a 0.25 µm CMOS technology and occupy 2 mm^2 and 0.8 mm^2 respectively.

An interference resilient 60 kbps-to-10 Mbps body channel transceiver for multimedia and medical data transaction in BANs was presented by Cho et al. [CYBY09]. The transceiver uses the human body as a signal transmission medium in the 30 MHz to 120 MHz frequency range for energy-efficient and scalable data communication around the body. In the frequency range of BCC, the human body may operate as a receiving antenna and pick-up large interferences, degrading its Signal-to-Interference Ratio (SIR) to 22 dB. In order to avoid body-induced interferences, a 4-channel Adaptive Frequency Hopping (AFH) scheme that monitors channel status continuously and selects only the clean channels was added to the BCC module. To support 10 Mbps wideband Frequency Shift Keying (FSK) signaling with the fast AFH, the direct-switching modulator and the DLL-based demodulator are incorporated in the transceiver design. The dual frequency synthesizers in the modulator reduce the frequency hopping overhead to 4.2 µs without degradation of the phase noise. The transceiver fabricated in 0.18 µm CMOS technology withstands a 28 dB SIR and operates at 1.8 m with 25 dB SIR.

An Impulse Radio (IR) type transceiver for HBC application has been presented by Shikada and Wang [SW12]. In their system, the transmitter employs an IR scheme in which the information data is converted to pulse trains. First, information with a date rate of 1.2 Mbps was encoded with eight pulses for bit “1” and nothing for bit “0”. The pulses were produced by a 4.9 MHz oscillator and an XOR gate, so the pulses had a duration of 10 ns. The modulated pulses were then spectrum-shaped by a band-pass filter to restrain their main spectrum components in the 30 MHz to 50 MHz range. The output signal drives an electrode which is placed on the body. The receiver employed the envelope detection scheme. The received signal was filtered, amplified, and then adjusted to an adequate level by an Automatic Gain Controller (AGC). The envelope detection is realized by a diode and a low-pass filter. The detected signal is then judged as “1” or “0” by a comparator after the envelope detector.

Magnetic Human Body Communication (MHBC) employing a quasimagnetostatic field was proposed by Ogasawara et al. [OSFM14]. The system is designed to mitigate the influence of the surroundings on the communication channel. It uses a human body as one component of a loop

that creates magnetic coupling between two transmitters. A circuit model was devised, and its validity was experimentally demonstrated. They have reported that the results show the validity of the MHBC channel model and its robustness with regard to the surrounding environment. They also show that HBC based on magnetic coupling is a feasible technology. The available data rate of the system is not reported.

A wearable wireless ECG sensor was presented by Nemati et al. [NDM12]. This system combined an appropriate wireless protocol for data communication with capacitive ECG signal sensing and processing. The ANT protocol was used as a low data-rate wireless module to reduce the power consumption and size of the sensor. Small capacitive electrodes were integrated in a cotton T-shirt together with a signal processing and transmitting board on a two-layer standard printed circuit board. Signal conditioning and processing were implemented to remove motion artifacts.

As referred earlier in Chapter 1, the wearable system presented in this dissertation consists of embedded sensors in textiles, all connected to each other with conductive yarns in a mesh topology. The main advantages of the proposed system, compared to previous works, are low energy consumption and the support for large number of sensors, while keeping the data-rate high. Two prototypes of the sensor nodes are presented in Chapters 4 and 6. The node-to-base communication in the second prototype (which is an improved version of the first) uses hybrid circuit and packet switching. Those packets delivered from sensor nodes to the base station can use circuit switching, while in the reverse direction, or between sensor nodes, packet switching is available. The communication module in the second prototype has been implemented on an ASIC with a 0.35 µm CMOS technology. The maximum data rate of the system is found to be 35 Mbps for tens of sensors.

2.7 BAN Operations by Network Layers

According to the Open Systems Interconnection model (OSI) architecture, a network consists of a stack of several layers, each with its own specific tasks. In this section, the physical, MAC and network layers of BANs are discussed. Since the proposed routing protocol presented in Chapter 3 can be used in both BANs and WSNs, the routing protocols for WSNs are also scrutinized in this section.

2.7.1 Physical Layer

The physical layer consists of the networking hardware and transmission technologies of a network. This layer provides electrical, mechanical, and procedural interface to the transmission medium. The choice of the physical layer depends on the application. It also depends on the sensor placement; in, on or off-body. This layer provides a procedure for transforming a Physical Layer Service Data Unit (PSDU) into a Physical Layer Protocol Data Unit (PPDU). Activation and deactivation of the transceiver and data transmission and reception are the main tasks of physical layer [MAL+14].

IEEE 802.15.6 specifies three different physical layers: HBC, Narrow Band (NB) and UWB [KUU10]. NB PHY is responsible for data transmission/reception, activation or deactivation of the radio transceiver and Clear Channel Assessment (CCA) in the current channel. NB PHY uses Differential 8-Phase-shift Keying (D8PSK), Differential Binary Phase-Shift Keying (DBPSK) or Differential Quadrature Phase-Shift Keying (DQPSK) modulation techniques (except in the 420 MHz to 450 MHz range, where it uses the Gaussian Minimum Shift Keying (GMSK) modulation technique).

HBC PHY defines the Electrostatic Field Communication (EFC) requirements covering modulation, preamble/Start Frame Delimiter (SFD) and packet structure. The PHY header consists of the following fields: data rate, pilot information, synchronization, WBAN ID, payload length and a CRC calculated over the PHY header.

The UWB physical layer is used for communication between on-body devices and for communication between on-body and off-body devices. Transceivers in a UWB PHY generate similar signal power levels to those used in the Medical Implant Communication Service (MICS) band and also allow low implementation complexity. Based on the specifications for UWB Physical Layer (PHY), the PPDU bits are converted into RF signals for wireless transmission. The UWB PPDU consists of a Synchronization Header (SHR), a PHY Header (PHR) and PSDU. The physical header carries information about the scrambler seed, length of the payload and the data rate of the PSDU. The receiver uses the information in the PHR to decode the PSDU [KUU10].

2.7.1.1 Radio Channels

Compatibility with worldwide regulations is a key issue for the selection of physical layer frequency bands for wireless sensors. UWB offers higher data rates and higher throughput, whilst lower frequencies have less fading, shadowing and attenuation from the obstacles and human body [BSM+12]. The HBC PHY operates in two frequency bands centered at 16 MHz and 27 MHz and has a bandwidth of 4 MHz. The European standard only supports only 27 MHz operating band, but the United States, Japan and Korea support both bands [MAL+14].

The NB PHY has seven different frequency bands and it offers various data rate, channels and modulation schemes. MICS is the first licensed band in NB PHY and is utilized for implant communication in the range from 402 MHz to 405 MHz. Wireless Medical Telemetry Service (WMTS) (used in medical telemetry systems) is the second licensed band in NB PHY. Both MICS and WMTS only support low data-rate communications. Usually for high data-rate applications, the ISM band is available. Nevertheless, because of various wireless devices such as IEEE 802.15.4 and IEEE 802.15.1 work in the same band, there is a high probability of interference [KUU10]. The frequency band from 2360 MHz to 2400 MHz) as the sixth band of the NB PHY is assigned for medical device use. It should be noted that the this band is not an ISM band; hence, in comparison with other ISM bands, interference is significantly reduced. The last band (2400 MHz to 2483.5 MHz) is a license-free ISM band, and for that reason is commonly used in a variety of applications [TDFT+10].

Figure 2.3: Conductive yarns (source: http://www.kobakant.at/DIY/?p=1978).

The UWB PHY defines two frequency bands: high band and low band. They are divided into channels with a bandwidth of 499.2 MHz. The low band has three with a central frequency of 3993.6 MHz and the high band has eight with central frequency defined at 7987.2 MHz [KUU10].

2.7.1.2 Conductive Yarns

In wired wearable systems the communication is usually established by conductive yarns. In the last decade, conductive fibres and yarns have drawn considerable attention [WA07b, PVM12]. There are various kinds of fibres and yarns, used in several but specific applications. Figure 2.3 depicts some conductive yarns. Textile materials made of organic polymers have very good insulation. Usually, electrical charge is accumulated on the surface of organic polymers due to weak electrical conductivity of polymers. To make electrically conducting textile materials, it is essential to prevent of the accumulation of electrical charge. For that, there are several methods that can be used:

- **Adding carbon or metals in different forms such as particles, wires or fibres:** Inserting carbon or metals such as copper, steel, silver, nickel in the form of wires, fibres, and micro or nano particles to the textile structure provides the required conductivity [NBA12]. Carbon fibres and carbon-filled fibres have good conductive properties, but they have some aesthetic problems. On the other hand, metal fibres and wires which can be inserted into textile structures exhibit high conductivity, but they have also some disadvantages such as weight and cost.

- **Using inherently conductive polymers:** Polyanyline, polyvinyl alcohol, polypyrrole and polyamide are inherently conductive polymers [SRG08]. These kind of conductive polymers are still in development. Amongst these polymers, polyanyline has good environmental, thermal and chemical stability. In comparison to other materials, polymers have many important advantages, but they are still rather costly. Usually they are usable for the applications where flexibility, low weight and conductivity are required.

- **Coating with conductive substances:** The third kind of conductive yarns is based on using coated conductive fibres, whose coating can be applied through various techniques [NBA12]. These are also used in many application areas. To produce highly-conductive fibres, a metallic or galvanic coating can be used, but adhesion and corrosion resistance and suitability of the substrate are some of their limitations.

By using conductive yarns in the fabric structures, various functionalities may be assigned to the fabrics, making them useful for many applications. In fact, conductive textiles have important applications not only in medical and military fields, but also in the fields of fashion, multimedia, gaming, architecture and design for fashion [PVM12]. Therefore, textiles with conductivity function are used in many technical applications such as BAN, WHS, protection of people and electronic devices from Electromagnetic Interference (EMI) and electrostatic discharge, heating, data storage and transmission, sensors and actuators. The work discussed in this dissertation employs conductive yarns for communication among sensor nodes (Section 4.1.1).

2.7.2 MAC Layer

In wireless and HBC BANs, the communication part of the sensor consumes most of the energy, which needs to be taken in strong consideration in the case of energy-efficient applications [UHB+12b, ANM+12]. The MAC protocol plays a significant role in controlling the communication module and in reducing the energy consumption, increasing throughput, and reducing delay. In a shared environment, such as radio communications, a collision occurs when more than one packet is transmitted at the same time. After collision, packets have to be retransmitted, which consumes extra energy and diminishes the throughput. The receivers have to continuously or periodically listen to the channel (even in idle mode), which also causes energy consumption. Usually, overhearing occurs, i.e. the received data may actually be for another node, which adds another detrimental effect on energy efficiency. An additional source of energy consumption is the control packet overhead, since control information must be added to the payload. Hence, a minimal number of control packets should be used for data transmission.

Regardless of the network type, MAC protocols are grouped into two contention-based and schedule-based protocols [GP10]. In contention-based MAC protocols such as Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), nodes contend for the shared channel to transmit data. If the channel is occupied, the node defers its transmission until it becomes idle. Although CSMA/CA protocols are scalable and have no strict time synchronization constraint, they impose a significant protocol overhead. In contrast, schedule-based protocols, such as Time Division Multiple Access (TDMA) protocol, divide the channel in time slots of fixed or variable duration. These slots are assigned to the existing nodes and each node transmits only in its own slot. Schedule-based protocols consume less energy than contention-based protocols. In fact, there are no contention, idle listening and overhearing problems. However, for appropriate operation, these protocols require frequent synchronization.

One of the most important attributes of a good MAC protocol for BAN is energy efficiency [USS+09]. In some applications, the device should support a battery lifetime of months or years without replacing or charging, while other applications may require a battery life of a few hours. For example, cardiac defibrillators and pacemakers have a lifetime of more than 5 years while swallowable camera pills have a lifetime of 12 h [KUU10].

QoS is also an important factor to be considered in good MAC protocols for BAN applications [KSTP12]. QoS also includes point-to-point delay and delay variation. In some cases such as fitness and medical surgery monitoring applications, real-time communication is required. The latency of data transmission depends on the application. For example, for multimedia applications, latency should be less than 250 ms and the jitter should be kept below 50 ms [UHB+12a]. For emergency applications, the MAC protocol should allow nodes to get very quick access to the channel (in a matter of few millisecond) to send the emergency data to the coordinator with minimal waiting time. Detection of irregular heartbeat, high or low blood pressure or temperature, and excessively low or high blood glucose level in a diabetic patient are some of examples.

One energy consumption reducing method for the MAC layer is the Low Power Listening (LPL) [CBS+11] (also known as channel polling). In this approach, nodes wake up for a short duration to check the channel activity. If the channel is idle then the nodes go into sleep mode, otherwise they continue to be active on the channel to receive the data. Usually, the sender sends a long preamble before each message in order for it to be detected by the polling at the receiver. Such a periodic sampling method is efficient for high-traffic nodes and performs well under variable traffic conditions but it is ineffective for low-traffic nodes, especially for in-body nodes, where periodic sampling is not preferred due to strict power constraints. In wired environment with baseband communication, receivers are always on. Because of the very low power consumption of receivers in their idle mode (a few µW in comparison with tens of mW for wireless receivers), such a periodical listening method is not necessary.

2.7.2.1 Related Work

This subsection is devoted to the analysis of reported work related with MAC protocols for BAN applications. The IEEE 802.15.4 is a standard MAC protocol destined to low data-rate wireless applications [AME06] and was designed to operate in three frequency bands: 868 MHz, 915 MHz and 2.4 GHz. IEEE 802.15.4 operates in two modes: beacon and non-beacon enabled modes. In the first the coordinator controls the synchronization of the device, association and data transmission by using periodic beacons. However, this mode of operation defined by the standard is not suitable for BANs due to its asymmetric transmission support. The Non-beacon-enabled mode of IEEE 802.15.4 uses un-slotted CSMA/CA. The un-slotted mechanism exhibits good performance in terms of bandwidth utilization and latency [LLR09]. Nevertheless in non-beacon-enabled mode, CCA may lead to high energy consumption that it should be taken into account.

In [SZ09], the authors propose a battery-aware TDMA based MAC protocol with cross-layer

design to maximize the network lifetime. The proposed protocol takes into account the following parameters for medium access: electrochemical properties of the battery, time varying wireless fading channel, and packet queuing characteristics. This protocol operates similarly to IEEE 802.15.4 beacon-enabled mode, where nodes listen periodically to beacons from the coordinator. To reduce energy consumption and increase lifetime, the nodes remain in sleep mode during idle periods. Reliable delivery of packets is achieved using this protocol, but there is no mechanism defined for emergency data. In addition, holding packets in buffers for long time leads to high average delay and packet drop rate.

The work in [LT10] also reports a TDMA based protocol for BANs but uses the heartbeat rhythm for synchronization. The network topology is star, where a central node is used to coordinate the network. Having the heartbeat-rhythm as reference, the Hybrid MAC (H-MAC) [GPOL05] is capable of providing the required synchronization for a TDMA approach that avoids the need of periodic control messages. This protocol leads to a reduced overall energy consumption. Each sensor extracts the heartbeat-rhythm information from its sensory data. It should be noted that heartbeat-rhythm is not accessible to all sensors (e.g., accelerometers). In such cases the devices cannot by synchronized. One solution is the integration of accelerometers with other sensors to facilitate the access to the heartbeat, which complicates the sensor design. In this method, the insertion of a guard-band time to avoid collisions leads to a decreased bandwidth utilization.

The protocol presented in [IA14] is a hybrid and secure MAC protocol called Priority Guaranteed MAC (PMAC) for BANs. This protocol uses two Contention Access Periods (CAPs) for accommodating normal and life-critical traffic and one Contention Free Period (CFP) for accommodating large amount of data packets. To prevent illegal access to the network, a set of security keys is used. The analytical and simulation results of the protocol indicate better performance than IEEE 802.15.4, in terms of power consumption and throughput.

The MAC protocol from [LPN$^+$10] works over a serial bus made of conductive yarns in a wearable system with no Carrier Sense Multiple Access (CSMA) function. In this protocol, the MAC layer uses a single-host node polling of other peripheral nodes. The MAC layer is able to provide both reliable and best-effort services. For the case of reliable communication, the protocol starts with a Token Frame (TF) transmission by the host and ends with a Transmission Confirmation Frame (TCF) reception by the host. If an error or loss of a frame occurs, the transmission of a frame is repeated up to three times, with 10 ms timeout interval to guarantee QoS. The resulting prototype showed that the MAC protocol can handle various types of network traffic, such as sensory, command, and multimedia data, as well as adapt to communication line faults due to the abrasion of conductive yarns.

The two prototypes of SNs presented in this dissertation utilize different MAC protocols. The first, presented in Chapter 4, uses CSMA/CA (Request to Send (RTS)/Clear to Send (CTS) handshaking) to share the bidirectional links. The second prototype in Chapter 6 is also based on RTS/CTS handshaking, but with the exception that RTS is also used to establish a circuit path over intermediate nodes between the sender SN and BS. As will be explained in Chapter 6, scheduling the transmissions of SNs over the circuit paths with a TDM protocol (cf. Section 6.1.1) achieves

much better performance in terms of power consumption, end-to-end delay and throughput in comparison with node-to-node CSMA/CA.

2.7.3 Network Layer

WSNs are by nature ad hoc networks in which techniques for network self-organization and routing of packets are generally employed [ASSC02]. However, there are many fundamental differences between traditional wireless ad hoc networks and WSNs, making conventional ad hoc network protocols unsuitable for WSN applications. Large number of sensors, proneness to failure, fast changing network topology, resource and power limitations are examples of such differences. The literature describes numerous protocol designs targeting specific WSN applications [IM05, Li08, ASSC02, DP10]. In a network using a reactive routing protocol, SNs acquire routes on-demand and avoid saving information about the network topology. Flooding, Gossiping and MCF [YCLZ01b, PDJ09] are the classic reactive protocols.

Energy efficiency usually is the first and most important design challenge for a WSN [DP10]. Energy-efficient protocols try to reduce the energy consumption by channeling the communication flow over the paths with minimum cost, increasing the lifetime of battery-driven SNs. However, traffic in sensor networks displays, in general, a heterogeneous nature [MM04]. In most cases two communication patterns can be distinguished: 1) traffic between the BS and sensor nodes, 2) traffic between adjacent nodes. The different characteristics of the two kinds of traffic need to be considered in designing a routing protocol for sensor networks.

A lot of research is being done towards energy efficient routing in ad hoc networks and WSNs [AY05, RAPR13], but most of the proposed solutions are inadequate for BANs. For example, in WSNs having high throughput and low routing overhead are more important than minimal energy consumption. Energy efficient ad hoc network protocols attempt to find routes in the network that minimize energy consumption in nodes with small energy resources, thereby ignoring many parameters such as the amount of operations and energy required to transmit and receive. Unlike BANs, in WSNs node failure is not problematic. In BANs the number of devices worn by a patient is smaller than the typical number of nodes in a WSN. On the other hand some WSNs protocols only consider networks with homogeneous sensors [DP10], which does not apply when considering different devices in BAN. In brief, although challenges faced by BANs are in many ways similar to sensor networks and WSNs, however there are intrinsic differences between the two, which require special attention in each context.

2.7.3.1 Related Work

Energy-efficient routing techniques for WSN can be classified in clustering and tree-based approaches [BB10]. In clustering protocols like LEACH [HCB00] the network is divided into clusters and cluster heads. Randomly selected cluster heads aggregate data from cluster members and transmit them to the base node. In tree-based protocols like PEGASIS [LR02] and MCF

[YCLZ01b] each node forwards data to a neighbour until the data arrives at the BS. In comparison to cluster-based protocols, tree-based approaches are more energy efficient. Tree formation in PEGASIS is based on a greedy algorithm that constructs a set of branches from the smallest possible constituent parts. Each node collects, fuses and forwards data to a designated neighbour. Although the routing algorithm tries to optimize the chosen tree, it does not always find the best solution; in fact, sometimes the worst possible solution may be chosen. PEGASIS is a simple and energy-efficient protocol, but it is not an optimal approach.

The MCF protocol is an energy-efficient method for routing packets in a reactive sensor network [YCLZ01b, PDJ09]. MCF routing is a cost-field-based approach that ensures that packets travel in one direction (from SNs to BS) with minimum communication cost. The cost field, which specifies for each SN the minimum cost to reach the BS from that node, is established during the setup phase. Each message specifies the minimum cost to the BS and the cost of the path traveled so far (the current path cost). A given SN forwards the message only if the sum of current path cost and the SN's cost is equal to the minimum cost specified in the message. Like PEGASIS, the SNs require neither routing tables nor information about the network topology, a significant advantage for resource-constrained systems.

This approach is only applicable to data sent from SNs to the BS. If the BS needs to send data to a specific node, other methods must be employed. For instance, flooding [PBBvR06] is used in some ad hoc routing protocols like AODV [PR99] or ODMRP [LSG02]. In situations where the BS generates a significant amount of data, flooding will reduce network performance [AY05]. Therefore, MCF is appropriate only for those applications where the BS acts almost exclusively as a data collector.

During the normal operation of the system, failures of links and nodes may occur frequently. Data messages will be lost if a node on a path fails, affecting all traffic that originates from nodes sharing a path containing the failed node. MCF does not employ any mechanism to recover from failures. It has been suggested that the BS could periodically refresh the minimum cost field by repeating the initialization [HT06]. During cost refreshing, data communications in the whole network are disrupted and additional energy is expended.

The MCF protocol may lead to multiple paths with equal cost. Messages received by different SNs with the same node cost will be duplicated and will result in similar data messages traversing the network over more than one path. This consumes unnecessary bandwidth and power, causing multiple copies of sensor messages to arrive at the BS. Serial numbers or timestamps have been proposed to avoid this message duplication [HT06]. The drawback lies in the need for having a table to store the serial number or time-stamp of the received messages at each node. This table may become large, especially for nodes close to the BS. The duplication problem is solved in this work by using unicast data communication between SNs.

The MCF method is not applicable to communication from the BS to a specific SN. One solution is to create a table with all minimum cost paths at BS and use that information to communicate directly with the sensors. This is the solution adopted in the present work (Chapter 3). Having routing path information only in the source node and avoiding routing tables in intermediate nodes

is the approach used by Source Routing (SR) in ad hoc networks [ZY03]. In this approach, the packet contains the address of each node in the routing path.

SR requires the determination of the address of all nodes and routing paths from source to destination as performed by protocols such as Dynamic Source Routing (DSR) [ZY03, GKK+03] for wireless ad hoc networks and Link Quality Source Routing (LQSR) [DPZ04] for wireless mesh networks. These protocols are reactive approaches and do not keep routing tables. Whenever the source node needs to send data, these protocols determine the route on-demand and keep the routing information only while communicating. Two disadvantages of DSR and LQSR are having higher connection setup delay than table-driven protocols and the absence of a mechanism for local repair of failed links.

In essence, the protocol proposed in Chapter 3 is based on the MCF. The drawbacks of MCF for communication from BS to SNs are avoided by implementing SR routing, so that all communication is done over paths with minimum cost.

2.7.4 Cross-Layer protocols

Cross-layer design is a way to improve the efficiency of and interaction between protocols by combining two or more layers from the protocol stack. This line of research has received a lot of interest in the field of sensor network [RCSA05, MVP06]. However, little research has been done for BANs. Ruzelli et al. proposed a cross-layer energy efficient multi-hop protocol built on IEEE 802.15.4 [GRMP07]. In this protocol, the network is divided into timezones where each timezone takes turns in the transmission. First the nodes in the farthest timezone start transmission. In the next time slot the farthest one sends its data and so on until the sink receives it. The protocol almost doubles the lifetime of the network compared to regular IEEE 802.15.4. Originally, the protocol was developed for regular WSNs, but it is also usefulness for BANs.

Cascading Information retrieval by Controlling Access with Distributed slot Assignment (CICADA) protocol is an improvement protocol [BBI+07]. CICADA is a cross-layer protocol specifically designed for BANs, based on a multi-hop TDMA scheduling approach [BBI+07]. It uses the same packets for both MAC as well as routing. The packets are used to detect the presence or absence of the children and to control MAC layer. First of all, the protocol sets up a spanning tree and divides the time axis in slots grouped in cycles in order to lower the interference and to avoid idle listening. Synchronization of time slots is possible because a node knows the length of each cycle. With the tree structure, each node informs its children when they are allowed to send their data to the sink. With CICADA, the network operates in converge cast mode, but traffic between individual nodes in the network is possible, as well by routing over the sink.

In Chapter 6 the circuit paths for communication between SN and BS in MAC layer (cf. Chapter 6) are provided by the network layer, which implies cross-layer operation. All the paths are selected to have minimum cost forwarding from SNs to BS as explained in Section 3.1.2.

2.8 Clock and Data Recovery

In any digital communication system, synchronization between the receiver and transmitter for data recovery plays a major role. Synchronization refers to the process of making events occur at the same time in different parts of a distributed system [Gib02, HS06, MMF98]. To perform successful Clock and Data Recovery (CDR), symbol or bit synchronization is done in every digital receiver by extracting timing information from the received signals. A decider circuit at the receiver uses that timing information to determine when to sample the incoming data. The optimum sampling instant is in the middle of the bit period. However, sampling precisely at that instant is not guaranteed, because of channel noise and jitter. Nonetheless, receivers can still correctly detect data by sampling within a small interval of time around the ideal instant. The sampling period depends entirely on the method used for clock recovery. Usually, keeping the sampling period in small ranges demands an increase in bandwidth to include more information for synchronization, and/or more complex receiver architectures.

Depending on the synchronization method, symbol synchronization is classified in two main groups [Gib02, HS06, Mad08]:

1. **Feedback or closed-loop:** In the closed-loop approach, the receiver attempts to lock a local oscillator to the incoming clock and generates a synchronized clock to recover the data. Usually, a Phase-Locked Loop (PLL) or a Delay-locked Loop (DLL) are used to track the incoming clock. The early-late gate synchronizer is one of the must widely used closed-loop methods [ZCP08a]. Like all feedback systems, closed-loop synchronizers need some time to lock and produce a stable signal at the output.

2. **Feedforward or open-loop:** In this approach, the receiver generates a synchronizaed clock without feedback. A popular approach is to generate the sampling clock by filtering the incoming signals with a band-pass filter, and feeding the resulting signal to a comparator, thereby creating a square clock signal. For this approach to work, the base-band modulation has to contain a spectral component at the data rate to enable the extraction of clock signal by filtering. Another method relies on oversampling the incoming data at a rate, which is at least 3 times higher than the data rate [tHS08, PCS$^+$08]. The recovered data value is chosen to be the value (“1” or “0”) according to the more often sampled in each data period. For the decision to be unambiguous, the number of samples must be odd. Oversampling is able to provide data recovery without delay.

CDR circuits based on closed-loop systems are more complex than open-loop systems, but they are widely used in many systems because of their good performance with low Signal to Noise Ratio (SNR) systems and signals. Nevertheless, in communication systems with relatively high SNR and limited resources, like sensor networks in wearable systems, an appropriately designed open-loop system may be preferable.

2.8.1 Related Work

Many CDR methods have been discussed in the literature. Descriptions of the fundamental aspects of synchronization and of basic CDR techniques can be found in [MMF98, Mad08, MD97]. In many open-loop methods the data signal carries spectral information about the clock signal. Filtering methods process the input signal and then use a band-pass filter to remove unwanted frequency components, followed by a high-gain amplifier and a Schmitt trigger to generate a square clock signal. A simple processing step is to rectify the signal by using a square-law device [BDMS13]. Another method multiplies the incoming signal with a 90° phase shift of the carrier signal, which is generated by filtering of the incoming signal [Bre02]. A third method uses an edge detector at the input to detect the transitions of the signal and generate spikes and recover clock from spikes [Bre02].

Filtering methods are easy to implement and exhibit no lock time problems. Their main disadvantages are unavoidable non-zero mean tracking, low performance in the presence of noise and the requirement to carry clock spectral information in the signal.

Oversampling CDR circuits implement another widely used feedforward recovery approach [tHS08, KWS+11, BNIA09]. In this method each received data-bit is sampled several times within its time interval. To properly recover the data, at least three (but often five or more for higher accuracy) samples per bit are necessary. All sampled data is saved in a First-in First-out (FIFO) memory. For sifting the data from the samples, a bit boundary detector finds the sampled bits at the data edge; then a data selector determines the appropriate samples near the middle of incoming signal and recovers the data. Oversampling methods result in fast and stable CDR circuits, but they need a high sampling rate and large FIFO memories [tHS08].

In more complex communication systems, feedback methods have been used to overcome the disadvantages of the filtering and oversampling approaches [MMF98, LL08, YCH+06, YCL06, KH14]. In these systems, a local Voltage Controlled Oscillator (VCO), placed in a closed-loop configuration, generates the desired clock signal. In many cases, a PLL or DLL is used to generate the clock signal at the receiver, which is synchronized with the incoming data stream and is used for recovering the data bits [HWZ+12, CZ07, ANI11, KLPS11, CcDgLs+10, CWJ11, CW09, WLS+06]. Lock speed depends on the design, with DLLs being usually faster than PLLs. Several DLL-based designs are compared in [GRAMN13], with the fastest method requiring at least 5 clock cycles to lock to the incoming signal.

The early-late gate synchronizer is another widely used close-loop CDR circuit [ZCP08b, SMJ13, JJB13]. This method performs two integrations of the received signal over two different periods of the symbol interval. The difference of integrals is used as a feedback signal to control the local oscillator.

Another approach works by injecting data directly in a ring oscillator [KLLJ13]. A VCO-based ring oscillator is used to generate four differential phases of the clock and a selector circuit is used to select one of the four phases as the recovery clock. This method is more complex than the open-loop methods but it achieves instantaneous clock and data recovery.

The open-loop synchronization method introduced in Section 5.1 is implemented by a low-complexity (small size) fast-lock synchronization circuit. Synchronization is based on the open-loop selection of the correct phase of the receiver clock synchronously with the incoming signal. The clock generator of the receiver is an autonomous oscillator set to operate at the same nominal frequency. The circuit lock time is at most one clock cycle, faster than all methods based on PLLs or DLLs.

2.9 Time Synchronization

A time synchronization method with enough accuracy is a necessary part of many wired or wireless sensors applications. Monitoring of the environment, localization, security, TDMA based protocols, coordinated sleep wake-up scheduling mechanisms and data fusion are some of applications based on time synchronization [RN10]. For instance, in gait analysis for evaluation and diagnosis of mobility impairments it is important to know the relative timing of several different surface electromyography signals and their time relation to the inertial information about the movements of the lower limbs. Otherwise, a healthy condition may be diagnosed as a disorder.

Each individual SN is usually equipped with an independent clock generator. The use of a crystal quartz oscillator ensures good local clock accuracy. Although this kind of oscillators have good stability (with variation around a few parts per million (ppm)), it is not enough to keep synchronization during the whole time of activity: since the clock generators are working independently, small drifts in clock generation lead to synchronization loss. By using a GPS receiver, SNs can achieve clock synchronization in the nanosecond range. However, sensors usually do not need such a high accuracy. On the other hand, using GPS receivers in resource limited systems is not always practical or possible (for indoor use, for instance). Another approach is to use a synchronization mechanism to overcome and compensate the clock drifts in each node relative to a clock reference [SY04, LC10, WCS11, RLK$^+$09].

The time synchronization problem has been studied extensively in all areas of networking [SY04, RN10, SBK05]. Many parameters affect the synchronization between the nodes. Limited power supply and unknown message delay in WSN are important challenges that make synchronization difficult in practice [LW11]. In addition, the latency of message communication also affects synchronization [RLK$^+$09]. The delay associated with the process of sending a message has four components:

1. **Send time:** delay for assembling a message and handing it over to the MAC layer.
2. **Access time:** delay for accessing the channel.
3. **Propagation time:** delay due to signal propagation between nodes.
4. **Receive time:** delay for receiving and processing message at the receiver.

Except for propagation time, which depends on the communication environment and may be highly deterministic, the other delay sources are non-deterministic in general. In order to

reduce the effects of delay on synchronization, in some protocols, such as Reference-Broadcast Synchronization (RBS), send and access delays are shortened by sampling and injecting time information into the message while transmitting [EGE02].

Many synchronization protocols use two-way message exchanges between pairs of nodes to estimate the time skew and offset [LE02, Mil91, GKS03]. In this approach, delay and offset are calculated by sending and receiving messages with timing information and measuring the round trip delay. Depending on the protocol, the exchange may be started by the clock source node or by the client node (i.e., the node whose clock is to be adjusted). Each protocol uses different mechanism for calculating skew and offset. A second approach uses only one-way communication [EGE02, LSW09]. In this case, the clock source sends timing information to client nodes and client nodes estimate skew and offset depending on the protocol.

2.9.1 Related Work

A classification of synchronization protocols based on synchronization issues and application-dependent features can be found in [SBK05]. In a master-slave approach, the master node is the time reference and slave nodes synchronize their clock with master, while in peer-to-peer protocols each node can directly get timing information from the other nodes. Failure elimination and flexibility are the advantages of peer-to-peer protocols, but their control is more complex. Because of network size and topology, it is not always possible for two nodes to communicate directly. In many networks, specially when the number of sensors increases, communication is set through multi-hop connections via intermediate nodes. In this case, synchronization must be performed by sender-to-receiver or receiver-to-receiver approaches. In the former, the receiver node synchronizes its clock with the sender and re-sends timing data to the next node after synchronization; in the latter, the sender broadcasts timing messages and receivers in its coverage area exchange messages among themselves instead of interacting with the sender.

The Network Time Protocol (NTP) is a well-known time synchronization protocol based on two-way message exchange [Mil91]. NTP is one the most used protocols, specially on the Internet, and is known as an effective, secure and robust protocol.

The NTP clients synchronize their local clocks to the NTP time servers by statistical analysis of the round-trip time. To achieve highly precise time synchronization, the time servers are equipped with or synchronized to atomic clocks or Global Positioning System (GPS) signals. The significant complexity of NTP makes its implementation in sensor networks difficult. In addition, the non-determinism in transmission time of WSNs can introduce large delays. Therefore, NTP is suitable only for synchronization in WSNs with low precision demands.

If a clock source broadcasts its time, adjacent nodes will receive the same timing message at approximately the same time. In this way, receiver nodes can obtain timing information that is subject to very little variability in its delay. This one-way method has been employed in the Reference Broadcast Synchronization (RBS) protocol [EGE02]. In this approach, a node is selected as a time reference node for all other nodes in the network. Each node records its local time immediately after receiving the message and compares it with the received time. This protocol utilizes

a sequence of synchronization messages from clock source node in order to estimate both offset and skew of the local clocks relative to each other.

The Timing-sync Protocol for Sensor Networks (TPSN) is a sender-receiver, two-way synchronization protocol [GKS03] and was designed specially for multi-hop networks. TPSN performs synchronization in two phases: level discovery phase and synchronization phase. In the first phase, a root node as clock reference elects and builds a spanning tree of the network. In the subsequent synchronization phase, nodes synchronize to their parent in the tree by exchanging messages at the initiative of the child node. If the root node or the network topology change, TPSN has to start again with the discovery phase. Redoing the level discovery phase after any changes increases the traffic and the network energy consumption.

The Flooding Time Synchronization (FTSP) is another one-way protocol for multi-hop ad hoc networks [MKSL04]. A root node periodically broadcasts a message with global time stamping information. Each receiver nodes estimates its clock offset and skew by combining both global and local time information. The root node is elected dynamically and periodically based on the smallest node identifier and is responsible for keeping the global time of the network. FTSP is an effective synchronization mechanism in multi-hop ad hoc networks, but [LSW09] shows that clock skew increases exponentially with increasing network diameter.

The Precision Time Protocol (PTP), defined by IEEE standard 1588 [IEE08a], is a hierarchical master-slave architecture for clock distribution and time synchronization with highly precise synchronization [LE02]. In this protocol, the clock server periodically sends synchronization messages. A client replies to the server with a message including the reception time of the first message from the server and the replying time. Then the server estimates the delay and offset and sends this information with another message. The client uses the information in the second message for local clock adjustment. To avoid or minimize the effects of clock drift on timestamping, client nodes periodically set their time to the server time. PTP is a highly precise synchronization protocol, which can ensure clock skew values in the sub-microsecond range.

Although, the aforementioned protocols are widely used to provide synchronization, they are not hardware-aware protocols. In other words, their performance depends on the implementation characteristics of different systems. The main challenge in synchronization is the estimation of the non-deterministic delay of data transmission due to the properties of the hardware. If it is possible to keep variations of the communication and processing delay at the hardware level at a minimum, then achieving synchronization with a simpler and more resource-efficient method is possible.

The time synchronization method presented in Section 5.2 is implemented by a fully digital circuit for one-way master-slave, hardware-aware, highly precise synchronization. The circuit is designed to perform synchronization in the MAC layer, so that the deterministic part of the clock skew between nodes is kept constant and compensated with a single message exchange. In each sensor node, the synchronization circuit provides a programmable clock signal and a real-time counter for time stamping.

2.10 Conclusion

The background and state of the art in the context of BAN systems including network protocols, communication types and architectures, node types, hardwares, clock and time synchronization methods, have been presented along the chapter. Each section has summarized the essential information and knowledge and also advantages and disadvantages of each technology or protocol for BAN applications. In all subjects, the latest related work has been introduced. The following chapters build on this information to explain the design decisions taken during the dissertation work. Since the wearable platform is built on a contention-based network, in which SNs handover packets via bidirectional links and intermediate nodes save and forward packets to the BS, in the next chapter the adopted routing protocol is explained.

Chapter 3

A Routing Protocol for Sensor Networks

This chapter presents and analyzes SRMCF protocol as a reactive, energy-efficient routing protocol for WSNs which is used also for the wearable sensor networks. This protocol is established on the basis of SR [ZY03, GKK+03] concepts for ad hoc networks, and MCF [YCLZ01b] methods for heterogeneous WSNs. Since the proposed concept combines SR with MCF, it is called the SRMCF protocol. SNs maintain no information about the network topology, but packets (from sensors to BS or vice-versa) always communicate over paths with minimum cost. In this approach, only the packets from the BS to SNs include routing information [ZY03, JJ01]. The proposed protocol is intended for application scenarios where the availability of limited resources requires the energy consumption to be kept as small as possible. Experimental and simulation results show that the protocol reduces energy consumption and increases the system lifetime, while preserving network performance. In the following sections the protocol is introduced and then compared with MCF. Then the implementation of the protocol in C and C++ language as well as in OMNeT++ and Contiki is presented. The generated codes by Contiki is programmed on TelosB motes and used to collect the experimental results. The contributions described in this chapter are the following:

1. Design and implementation of the energy-efficient SRMCF protocol;
2. Analysis of SRMCF protocol packet count, packet header length and routing table size;
3. Experimental evaluation of SRMCF and MCF based on implementations for TelosB motes;
4. Extensive comparative simulations of both routing protocols running with two different MAC protocols.

3.1 Overview of the SRMCF Protocol

SRMCF is a reactive, energy-efficient routing protocol, where SNs maintain no information about the network topology, but packets (from sensors to BS or vice-versa) always communicate over paths with minimum cost. The SRMCF protocol specifies that SNs use MCF as the routing algorithm for sending acquired data to the BS. Communication in the other direction (from BS to

a specific sensor) over a minimum cost path typically requires intermediate nodes to keep stored information about minimum cost routing paths. But as mentioned in Chapter 2, nodes in reactive networks do not store routing information. Therefore, SRMCF requires that packets generated by the BS include the path information, like in DSR [AAWA08]. The intermediate nodes use the path information in the header to route packets without having to maintain information about the destination node. The BS must preserve a routing table of minimum cost paths from itself to the sensor nodes. When the BS needs to send data, it generates a unicast packet, adds the path information from its routing table to the header of the packet, and sends it to the first SN in the path. Intermediate SNs use the path information in the header to subsequently route the packet.

By combining minimum cost forwarding and source-based routing, a reactive protocol can be energy-efficient, providing optimum routing for both communication directions (to and from the BS), while taking into account the heterogeneous traffic characteristic of a WSN. Using minimum cost paths for communication in both directions minimizes power consumption.

3.1.1 Supported Message Types

The routing algorithms for packets coming from the BS node and for packets generated by sensor nodes are different. As a consequence, the header of data packets is also different. In both cases data transmission is unicast and packets are sent over minimum cost paths. Intermediate nodes select the appropriate routing algorithm based on the packet type. In addition, SRMCF supports broadcast messages for network setup and failure recovery.

3.1.1.1 Broadcast Messages

This protocol supports two kinds of broadcast messages: cost advertisement and cost request. Whenever the BS starts up or any SN gets a new cost value, they send a cost advertisement message to inform neighboring nodes of their current cost value (cf. Section 3.1.2). Nodes can broadcast cost request messages whenever a new cost value is needed. This occurs when a sensor turns on or a failure happens (cf. Section 3.1.3).

3.1.1.2 Unicast Messages from BS to Sensor Nodes

Figure 3.1 shows part of a WSN that includes the BS node and several sensor nodes. Each sensor has a predefined unique hardware ID address that is usually used as a MAC address. A sensor network may use the hardware identifier for both MAC and network address or extract the network address from the MAC address [YP09], a choice that depends on the application and the sensor network interface. Here the address is a 48-bit IEEE MAC which is used for most IEEE 802 network technologies, including Ethernet, Bluetooth and WiFi.

Suppose that the BS sends a packet to sensor N3 over the minimum cost path shown in bold in Fig. 3.1. For this type of packets, the header includes three fields for routing purposes: packet type, length and path. Each node uses the packet type to select the appropriate routing algorithm. The length field specifies the length of the packet in bytes including packet header and payload.

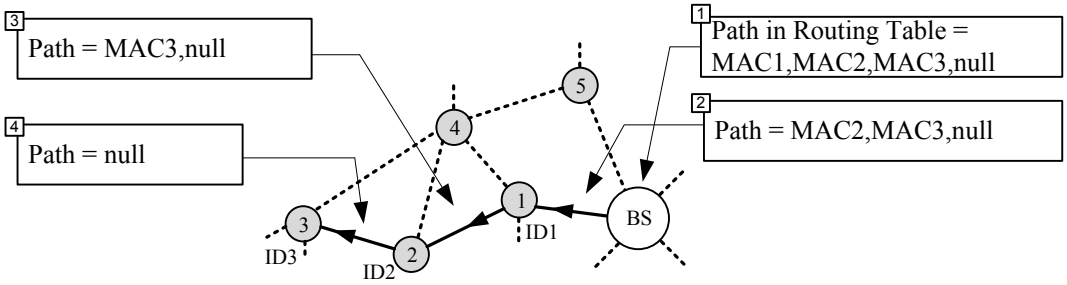


Figure 3.1: Value of path and length fields at different nodes on the path between BS and node SN3.

Path information consists of the MAC address of the intermediate and destination nodes, plus a final byte with null value. The initial value of this field comes from the routing table of the BS. Each intermediate node inspects the first byte of the path to determine the MAC address of the next node. Before routing the packet to the next node, each node eliminates the first byte of the Path field, decreases the length field by one, and encapsulates the packet. When the packet arrives at the destination node, the Path field contains only the null value. Since most of the power consumption comes from transmission and not from packet processing, elimination of the used path information decreases the average size of the header and the power consumption.

Table 3.1 shows the paths from BS to nodes SN2, SN3 and SN4 as present in the routing table of the BS. The entry for each node contains an ordered list of intermediate MAC addresses. Figure 3.1 shows the values of the path and length header fields as the packet passes through different nodes on the path from BS to SN3. This example assumes that the packet carries a payload of 200 bytes.

3.1.1.3 Unicast Messages from Sensor Nodes to BS

In this case the header comprises fixed-length fields for packet type, length, and ID of the source node. During the setup phase of the network (cf. Section 3.1.2) each node is assigned a minimum cost value, together with the ID and MAC addresses of the adjacent node on the minimum cost path to the BS (hereafter called the *near-node*). In this example, SN2 is the near-node of SN3, and

Table 3.1: Contents of the routing table in BS for paths to nodes SN2, SN3 and SN4 of the network in Fig. 3.1.

Num	Node	Path
...	...	...
2	ID2	MAC2, MAC1
3	ID3	MAC3, MAC2, MAC1
4	ID4	MAC4, MAC1
...	...	...

SN1 the near-node of SN2. Suppose now that the sensor SN3 in Fig. 3.2 needs to send a packet to the BS. SN3 generates a packet that includes its own ID and sends it to SN2, which passes the packet to SN1. Then SN1 sends it directly to the BS, which uses the ID field to identify the packet's source node.

It may happen that two nodes receive a packet with the same cost value. With MCF both receiver nodes will route the packet to the BS direction. Therefore, the BS will receive two copies of the packet. Obvioulsy, unnecessary messages decrease the network performance. Unicast communication between a node and its near-node in SRMCF avoids the problem with equal-cost paths: except for the near-node of each sender, all nodes discard unicast packets from that sender. So, with SRMCF message duplication due to equal cost paths never happens.

3.1.2 Network Setup

The setup phase has two steps. In the first, the cost field is set up: all nodes determine their cost values for communicating with the BS. In the second step the BS node creates the routing table. The setup processing is as follows.

3.1.2.1 Determination of Node Cost Value

This step is similar to the minimum cost forwarding back-off algorithm [YCLZ01a]. Each node sets its initial cost value to ∞ and BS to zero. Then the BS starts broadcasting its cost value. A receiver node compares its own cost value with the received cost (including the cost value of the sender plus the cost of the link between the sender and receiver). If the received value is lower than the node's current cost value, then the node updates its current cost to the new value, waits, and then broadcasts the new value. The waiting time increases linearly with the link cost value [YCLZ01b]. This process goes on until all nodes set their cost values to the minimum.

The example in Fig. 3.2 shows that node SN3 has two links with the same cost (to SN2 and SN4), but the cost value from SN2 is lower than the cost value from SN4 (3+2 $<$ 5+2). Therefore, SN3 will change its own cost to 5 and designate SN2 as its near-node.

The back-off algorithm decreases the number of cost advertisement messages significantly, as most of the nodes will broadcast their cost value only once. SRMCF utilizes a similar method but here the nodes hold a unique ID.

3.1.2.2 Routing Table Creation

The routing table in the BS holds information about all optimum paths (i.e, paths with minimum cost value) between the BS node and all other nodes. During the first phase the cost value of each node changes (from the default value of ∞ to the cost determined in that phase). When a node changes its cost value, it sends a packet with its ID and MAC address to its near-node. The receiver adds its own MAC address to the received payload and sends it to its own near-node along the optimum path. Eventually, the BS receives a packet that includes the ID and MAC addresses of the source node, and the MAC of all the nodes in the minimum path between the source node

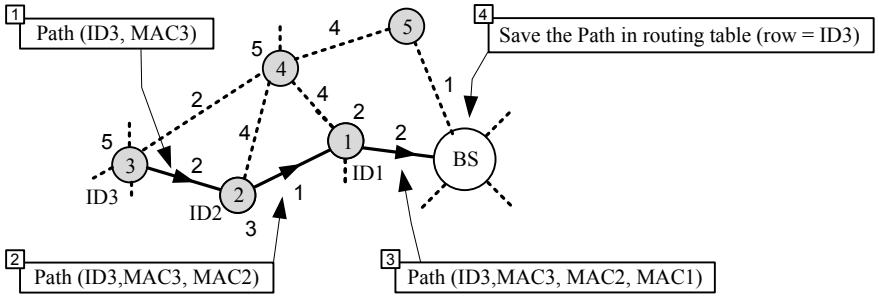


Figure 3.2: Adjusting the path from SN3 to BS when the cost value of SN3 has changed.

and BS. It then saves the ID and MAC addresses in the row of the routing table corresponding to that particular source node (cf. Section 3.1.1.2). Figure 3.2 shows the routing path creation for node SN3 and Table 3.1 shows the value of the row corresponding to SN3 (ID3 in the column "Node" of the routing table).

In this way, sensor nodes and BS collaborate in the creation of the routing table. In fact, the routing table is created without running any algorithm or calculation on the BS and without any information about the network topology. Note that the same procedure is performed during normal network operation whenever the cost value of a node changes. This may happen when a link or node failure occurs, or when a node gets a cost advertisement message with a lower value than its own previous cost.

3.1.3 Link and Node Failure Recovery

Recovering from a link or node failure involves updating the cost value field and the corresponding minimum-cost paths stored in the BS. In order to detect failures each node has a timer, which is restarted whenever the node gets a message from its near-node. Each node monitors the activity of its near-node. Any message received from that neighbor will reset the timer. If there is no message for a specified time period, the node sends a query message to check the reachability of its near-node. If there is no reply, the node initiates the recovery procedure by broadcasting cost request messages periodically.

Figure 3.3(a) shows a part of the network with established links and corresponding minimum cost paths (bold lines). In Fig. 3.3(b) node 1 has failed, and nodes 17, 13, 18, 14 and 16 must find new minimum cost paths. Figure 3.3(c) shows the network after recovery and Table 3.2 presents the cost values before and after failure. Node 13 selects node 12 as its near-node, because the new cost value of node 17 is higher than that of node 12. In general, the cost values of nodes which are closer to the BS than the failing node are not affected.

The failure recovery mechanism of SRMCF is mainly local, with changes restricted to the place of occurrence without affecting other parts of the network, a behavior that enhances energy efficiency. In addition, data communication is not disrupted during failure recovery, as happens when a periodic setup phase is used with MCF [HT06].

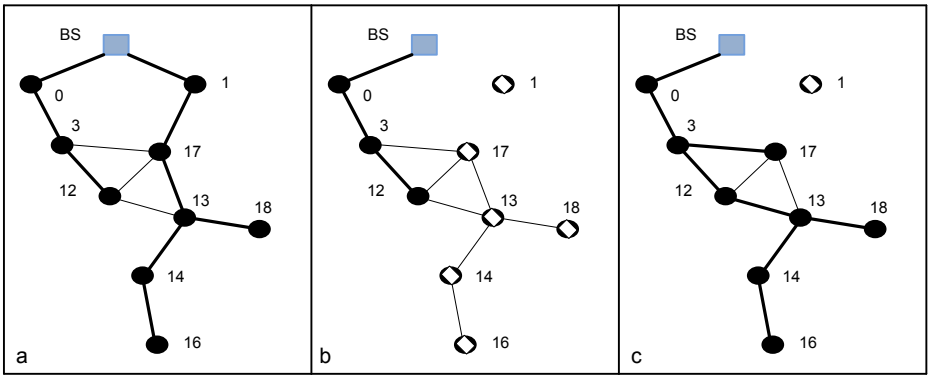


Figure 3.3: Failure recovery: (a) initial situation, (b) failure of node 1, (c) recovery from failure complete.

Table 3.2: The cost value before and after failure event

Node	**Cost**	
	Before failure	**After recovery**
0	55	55
3	102	102
12	154	154
17	139	156
13	192	199
18	244	251
14	240	247
16	286	293

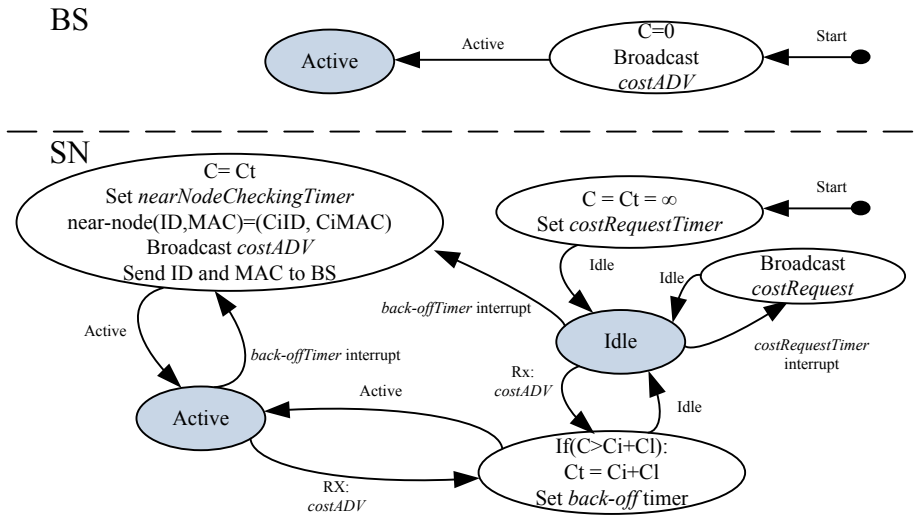


Figure 3.4: Network setup: determination of node cost values and initialization of BS.

3.2 Modeling and Implementation of the SRMCF Protocol

This section describes a C and C++ implementation of the SRMCF protocol for the TelosB mote platform [Cro13]. The generated codes are used for the protocol evaluation by simulation with OMNeT++ and also implementation of the protocol on TelosB mots by Contiki. The simulation and experimental collected results are described in Section 3.4.

3.2.1 Modeling of SRMCF Protocol

In the following subsections, modeling of the setup phase and nodes operation modes, routing operation and failure recovery of SRMCF are presented respectively.

3.2.1.1 SRMCF Setup

Figure 3.4 shows the Finite State Machines (FSMs) that describe the operation of SNs and BS in the first step of the setup (cf. Section 3.1.2.1). In this figure, C, C_i, C_l and C_t are respectively the current node cost, the node cost in the received message, the link cost and the temporary node cost.

The BS has only one mode of operation: after sending a cost advertisement message *costAdv*, it goes into active mode (normal operation). In contrast, each SN has three operation modes: idle, pause and active.

1. **Idle:** When a sensor is switched on, the register values for routing operations are undefined, and so the node is in idle mode, not participating in network communication. In this mode, a SN periodically sends a cost request message *costRequest* and waits for a *costAdv* message. All messages except those of type *costAdv* will be discarded.

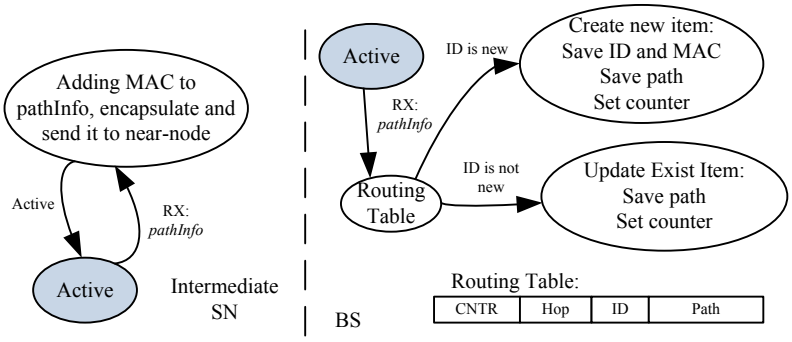


Figure 3.5: Routing table creation in the BS with collaboration of sensor nodes.

2. **Pause:** This mode is entered when a failure occurs (cf. Section 3.1.2.2). The node temporarily stops normal activity and waits for a reply from the near-node. If the near-node indicates its presence, the node changes to active mode; otherwise it goes to idle mode.

3. **Active:** After setting the cost value and registering the ID and MAC addresses of the near-node, the SN switches to active mode, in which it acts as a router in the network.

It should be noted that each SN has the ability to update its initial routing parameters when a *costAdv* message is received, even in active mode. Unlike what happens when the MCF protocol is used, routing operation of the node does not stop during the updating of the routing information.

Figure 3.5 shows the FSMs that describe the initialization of the routing table in the BS (cf. Section 3.1.2.2). The BS first checks the routing table to determine whether any previous information exists. If so, the BS updates the entry of the routing table; if the message is from a new node, the BS creates a new entry in the routing table and records the information.

Each entry of the routing table has four fields: *CNTR*, *Hop*, *ID*, and *Path*. Field *Hop* is a hop counter and determines the number of intermediate nodes between BS and the SN (length of *Path*). Field *CNTR* is a counter that is used to check for the presence of SNs and refresh the routing table. The meaning of fields *ID* and *Path* is described in Section 3.1.

The BS sets *CNTR* to an initial value and starts counting down when a new entry in the table is created. When any type of message from the corresponding SN is received, the BS resets the corresponding *CNTR* to the initial value. When *CNTR* reaches zero, the BS eliminates the corresponding entry from the routing table. The absence of messages from a SN in a specific time interval is taken to mean that the SN is not active or that it does not participate in the network. In this way, the BS eliminates unnecessary information from the routing table.

3.2.1.2 SRMCF Routing Operations

Figure 3.6 presents the FSM for a regular routing operation of the sensor nodes. The handling of the unicast packets is based on the packet type. All node-to-BS packets are forwarded to the near-node, and all node-to-node packets are handled by the receiving node. Neighbouring nodes communicate with each other using node-to-node packets to exchange information about their

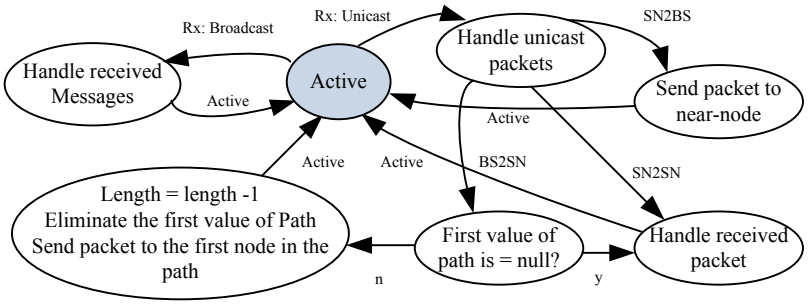


Figure 3.6: Routing operation of sensor nodes.

status. The destination of BS-to-node packets must be checked as well as the packet forwarded to the next node in the path field of the packet. The BS, as an end-point, accepts node-to-node and node-to-BS packets, but it is not involved in routing. Broadcast messages are handled regardless of the source, and rebroadcast if necessary. It should be noted that in active mode, the handling of messages can change the operation mode. For example, a cost request message received from the near-node resets the node to the idle state. Such a message means that the path with minimum cost is not valid any more, and the node needs to find another path.

3.2.1.3 SRMCF Failure Recovery

In any kind of network, faults occur frequently and unexpectedly [YMM07, PH07]. Usually WSNs are more prone to failure than traditional networks because of the limited resources. Any type of fault decreases the performance of the network and may even prevent all communications. Before recovery from a failure can be attempted, a detection mechanism must identify the abnormal situation. Many research efforts have been made to recognize, detect and recover from failures [YZW+11, BCK11, AYB13]. The method used here is based on self-detection [AMM08, YJY11] and near-node coordination [SHH05].

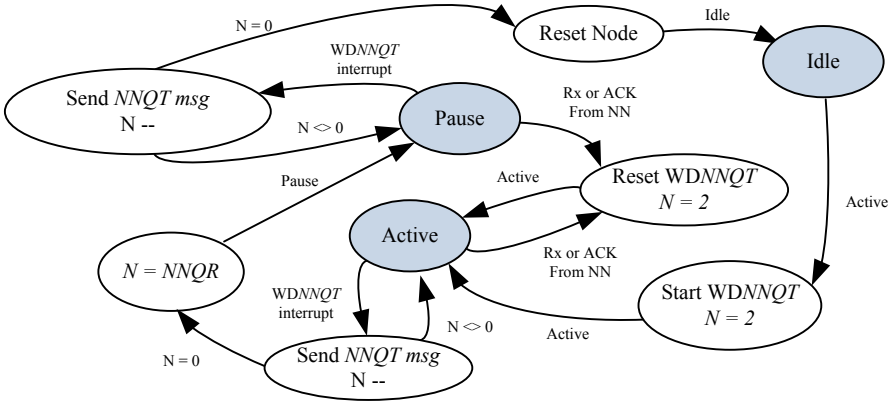


Figure 3.7: Failure recovery.

Figure 3.7 shows the failure recovery process in SNs. Failure control runs in the active and pause modes. A sensor node in active mode periodically monitors the activity of the near-node. Monitoring is based on messages received from the near-node (regardless of source, destination or type) and on sending query messages. Failure detection is done in two steps:

1. In active mode each node keeps a watchdog near-node query timer *WDNNQT* with interval T_q. The timer is reset when the node receives any messages from the near-node. If there is no message from the near-node in T_q seconds, the node remains in active mode and sends a query message *NNQ* in order to test for the presence of the near-node. If the inquiring node does not get any response, then it switches from active mode to pause, and starts the second recovery step.

   Remaining in active mode during the first step prevents a mistaken failure detection. For instance, the absence of messages from the near-node may be due to collision or interference. If the node starts the recovery procedure immediately, the effort and communication disruption caused by the recovery procedure may be useless. During the first recovery step, the node works normally and all processes are active.

2. The node is in the pause mode, and repeatedly sends *NNQ* messages to the near-node. The *NNQ* messages are sent up to *NNQR* times (with interval T_q), until an answer is received. In this step, any message from the near-node changes the mode to active mode. After *NNQR* messages without any response, the node assumes that the near-node failed. It then switches to idle mode, resets all internal routing information, and starts to gather new information from the network by sending a *CostRequest* message.

Determination of T_q and *NNQR* depends on network density, traffic and energy consumption. Small values of T_q lead to faster failure detection, but increase energy consumption and decrease data rate, because more *NNQ* messages are issued. Smaller values of *NNQR* increase the chance that temporary absence of communication over a link (for instance, due to collisions) may be mistaken for a node failure. The impact of the recovery procedure depends on the relative location of the SN. Nodes closer to the BS are on more minimum-cost paths than nodes on the periphery of the network. Therefore, the recovery procedure may have a large impact on the overall network performance. For outlying SNs, failure recovery only has local impact.

3.2.2 Protocol Implementation Details

Two different platforms have been utilized for simulation and implementation: OMNeT++ for simulation and Contiki for both simulation and implementation the protocols on the TelosB mote. The implementation details are described next.

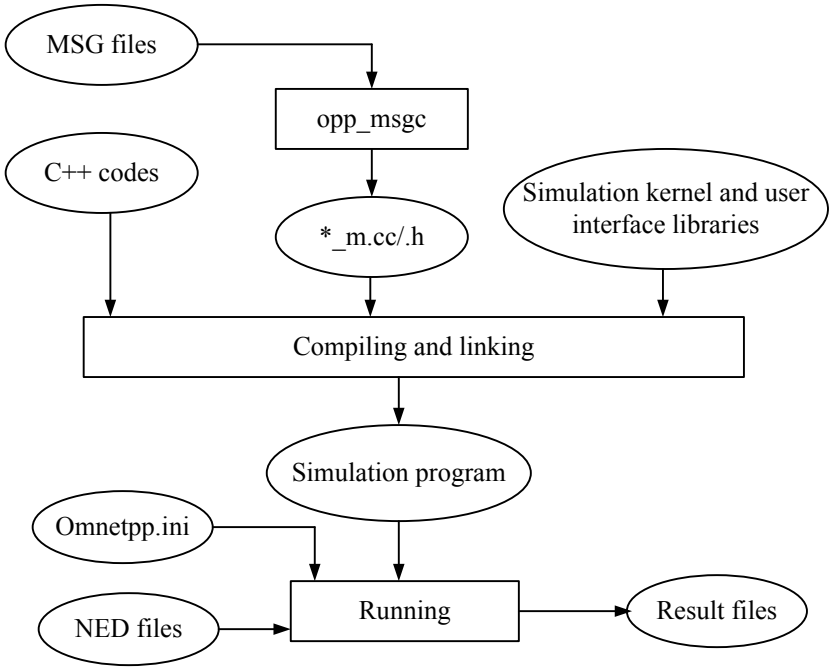


Figure 3.8: Building and running simulation in OMNeT++.

3.2.2.1 Implementation with OMNeT++

OMNeT++ is an open-source, extensible, modular, component-based simulation package built on C++, primarily for building network simulators for both wired and wireless communication networks [OMN11a]. In fact, OMNeT++ covers all kind of networks such as mobile, satellite, optical, on-chip networks, queuing networks, and so on. In order to meet the needs of different networking fields (such as sensor networks, wireless ad hoc networks, Internet protocols, performance modeling, photonic networks, etc.) domain-specific functionality is provided by model frameworks like MiXiM [MiX11, KSW+08], which are developed as independent projects. OMNeT++ includes an Eclipse-based IDE, a graphical runtime environment, and many other tools. There are extensions for real-time simulation, network emulation, alternative programming languages (Java, C#), database integration, SystemC integration, and several other functions. Figure 3.8 shows the work flow for building and running a simulation in OMNeT++ [OMN11b]. To build an executable simulation program, the MSG files need to be first translated to C++ using the message compiler (opp_msgc). All C++ sources need to be compiled into object files and all object files need to be linked with the necessary libraries to get an executable or shared library.

MiXiM is an OMNeT++ modeling framework designed for mobile and fixed wireless networks, such as wireless sensor networks, body area networks, ad hoc networks, and vehicular networks. It includes models of radio wave propagation, interference estimation, radio transceiver power consumption and wireless MAC protocols (e.g., IEEE802.11, Bluetooth and Zigbee). It is a merger of several predecessors OMNeT++ frameworks developed to support mobile and wireless networks.

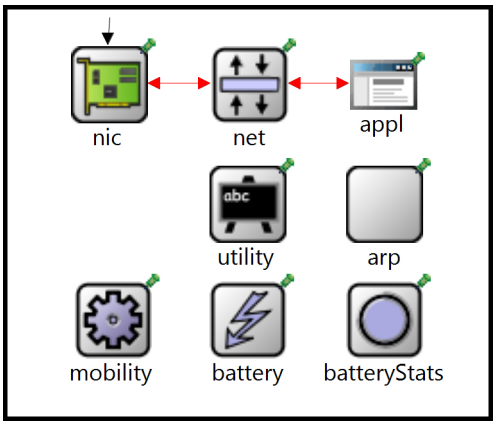


Figure 3.9: BS and SNs nodes in OMNeT++.

Any OMNeT++ model consists of the three following parts:

1. **Network description:** Network topologies and also all the nodes in OMNeT++ are described in the Network Description (NED) language. These are files with the "*.ned" extension.

2. **Message definitions:** Each layer of the network (such as MAC or network layers) has its own message structure. That structure is described in a file with ".msg" extension and *opp_msgc* message compiler generates compiled C files for use as library codes.

3. **C++ codes:** Include simple modules and any other C++ code in ".cc" files (or ".cpp" in Windows). These file are the set of the functions based on the user defined protocol. In fact, these files containe the codes to implement Figs. 3.4 to 3.7.

Figure 3.9 depicts a node with all submodules for either BS or a SN. The submodules communicate with each other by exchanging messages. The implemented submodules are the following:

1. **nic:** This submodule contains both physical and MAC layers. The physical layer of the employed motes (MICAz and TelosB) is based on the CC2420 IC, which is a single-chip 2.4 GHz IEEE 802.15.4 compliant RF transceiver [Tex11]. The MiXiM frame work contains all the libraries for the aforementioned IC. The MAC layer is implemented on top of the physical layer, and it can be any user defined protocol which is available.

2. **net:** The net submodule represents the network layer and includes all the code for routing and buffering the packets (the files *BSNetwLayer.cc* for BS and *NodeNetwLayer.cc* for the SNs). Any packet received by the nic submodule is sent to the net submodule for decapsulation and further processing. In the same way, submodule net sends packets down to the nic submodule.

3. **appl:** The application layer contains all the codes for running the other layers above the network layer. This submodule generates the traffic (*BSApplLayer.cc* and *NodeApplLayer.cc*) for simulation.

4. **utility:** The utility sub-module is used for statistical data collection and inter-modular data passing (status, location, energy level etc.).

5. **arp:** This submodule represents the Address Resolution Protocol (ARP), which is used by MiXiM to declare the addresses in the MAC and network layers.

6. **mobility:** The mobility submodule is responsible for the location of the nodes and also for their movements (if they are mobile). The nodes in the networks studied here are fixed and this module only determines their location. Location data is necessary to find the nodes that are within coverage range of other nodes.

7. **battery:** The battery submodule is used for modeling the energy reserve of the nodes and the energy consumption activity of nodes (including communication, processing, mobility-related energy consumption, etc.).

8. **batteryStats:** This module is used to collect data about battery status and to measure the energy consumption of the nodes. The collected data can be use for statistics analysis and also to determine the lifetime of the nodes and of the entire network .

After creating the NED files of the BS and SNs, the submodules have to be programmed in C++. After this step, the process is the same as for building any C/C++ program from source: all C++ sources need to be compiled into object files (.o files) and all object files need to be linked with the necessary libraries to get an executable or shared library. The executable file together with "omnetpp.ini" and the network NED file are used to run the network and get the results. The file "omnetpp.ini" is the network configuration file, which specifies all nodes and network settings.

3.2.2.2 Implementation with Contiki

Contiki is a lightweight open source operating system for the Internet of Things, which was originally designed to connect tiny low-cost, low-power microcontrollers to the Internet, but it can also be used for implementing any other kind of network [Con12, DGV04, ODE+06]. Contiki runs on a range of low-power wireless devices, such as TelosB and MICAz. It has been designed for tiny systems, having only a few kilobytes of available memory, capable of operating in extremely low-power conditions, such as wireless sensors. The Contiki OS follows the modular architecture shown in Fig. 3.10 [KKP09, FK11]. In the Contiki OS, all facilities such as, senor node data handling, communication, and device drivers are provided in the form of services. Each service has its own interface and implementation. Applications using a particular service just need to know the service interface and are not concerned with the implementation.

Cooja is the Contiki network simulator. Cooja contains a GUI and allows large and small networks of Contiki motes to be simulated and emulated, before being put in hardware. Motes can be emulated at the hardware level, which is slower, but allows precise inspection of the system behavior, or at a less detailed level, which is faster and allows simulation of larger networks. Cooja is a highly useful tool for Contiki development, as it allows developers to test their code and

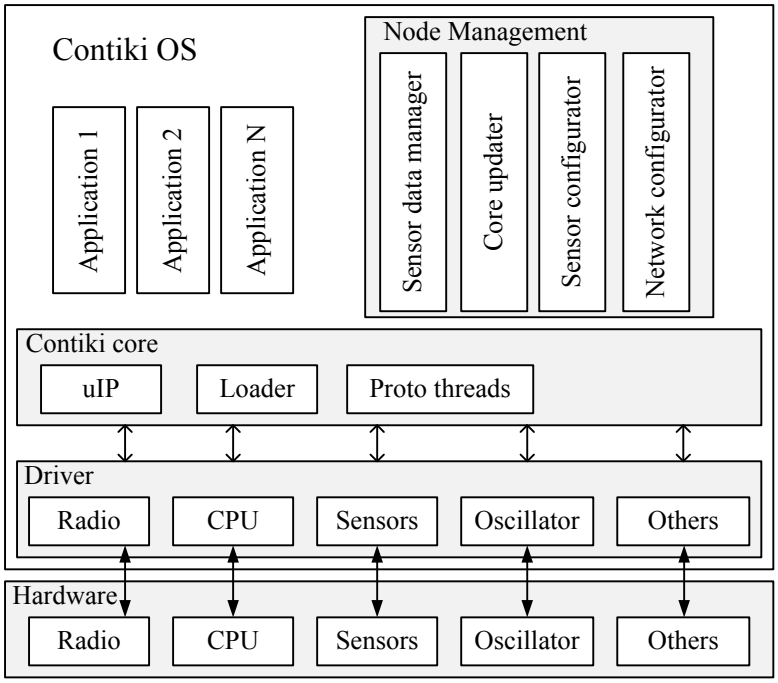


Figure 3.10: The architecture of Contiki OS.

systems long before running it on the target hardware. Developers regularly set up new simulations both to debug their software and to verify the behavior of their systems.

The C codes necessary to use in Contiki have almost the same structure as those created for OMNeT++, with some changes to be compatible with the Contiki environment. It should be noted that at the beginning of making a project in Contiki, the target mote has to be selected. For each supported mote, Contiki automatically generates from the code used for simulation an appropriate version to run directly on the mote. So, after compiling and running the simulation in Cooja, Contiki can upload the compiled code directly to the mote via a USB port.

3.3 Protocol Analysis

As mentioned, to increase the network performance and reduce energy consumption, SRMCF generates less packets than MCF for transmitting the same amount of data. This section analyzes SRMCF and compares it with MCF. In addition, average packet header length and routing table size are examined.

3.3.1 Packet Count

Let the set $N = \{n_1, n_2, ..., n_m\}$ denote the WSN nodes. If n_j is within the transmission range of n_i with reliable communication, then $l_{i,j}$ denotes the link between these nodes. If the propagation channel between nodes is reciprocal and nodes are equipped with the same transmitter and receiver circuits, then $l_{i,j} = l_{j,i}$. In this case, let L denote the set of such reciprocal links. Then we can model

the network as an undirected graph $G = (N, L)$. In the following, we assume that G is connected. Between each pair of nodes there are many possible paths. Let $P_{i,j}^k$ denotes the k-th path between n_i and n_j. For random networks with n nodes, both the maximum and the average number of hops in paths between nodes increase as $\Theta(\sqrt{N})$ [AER08, SMSR02].

In order to model the communication costs, we can assign positive, additive cost functions to the nodes and the links. Let $C_N(i)$ define the communication cost value of node n_i and $C_L(i, j)$ the cost value of link $l_{i,j} = l_{j,i}$. Let $C_{i,j}^k$ represent the cost for sending a message from node n_j to n_i over the k-th path. Then:

$$C_{i,j}^k = \sum_{r,s \in P_{i,j}^k} C_L(r,s) + \sum_{n \in P_{i,j}^k} C_N(n) \tag{3.1}$$

Since we are assuming reciprocal links, we have $C_{i,j}^k = C_{j,i}^k$, so the cost of communicating between two randomly selected nodes is independent of the direction.

Whatever the metric used to measure the cost, there is at least one path with minimum cost between two arbitrary nodes:

$$C_{i,j} = min\{C_{i,j}^k | k \in P_{i,j}\} \tag{3.2}$$

Obviously, the minimum cost value depends on the cost metric. If energy is used as the cost metric, Eq. (3.2) specifies a path that results in minimum energy consumption during communication; if the cost metric is the hop count, Eq. (3.2) specifies a shortest path. The SRMCF protocol uses the cost-field method described in [YCLZ01b] for calculating all $C_{i,j}$.

In a scenario where a sink node collects data from all other nodes, the total transport capacity or throughput of the sink is upper bounded by the maximum channel capacity W [MDMLN03]. In other words, if each sensor generates an equal amount of data to send to the sink node (BS), then the maximum throughput of nodes is limited to W/N.

Let λ_n and λ_s denote, respectively, the achievable throughput of each node and the total transport capacity of the sink node. If each packet must be sent k times on the path from sender to the receiver node, then λ_n can be at most $W/(krN)$. In this case, λ_s is upper bounded by:

$$\lambda_s \leq \frac{W}{r}. \tag{3.3}$$

Therefore, with increasing r the maximum achievable transport capacity of all nodes, including the sink node, will decrease.

Let T_M^n and T_M^s (M stands for MCF protocol) denote the total number if packets generated by nodes and sink node, respectively, when using the MCF protocol. Since the average path length increases as $\Theta(\sqrt{N})$, the total throughput of the nodes is

$$T_M^n = h\sqrt{N}\lambda_n N, \tag{3.4}$$

where $h\sqrt{N}$ denotes the average hop count and h is a positive value that depends on the network

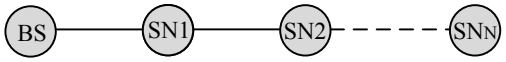


Figure 3.11: A network topology with maximum average hop count

topology. If the sink node communicates with $1/m$ of the sensor nodes, then

$$T_M = T_M^N + T_M^S = h\sqrt{N}\lambda_n N + \frac{\lambda_n}{m}N, \qquad m > 1, \tag{3.5}$$

where T_M denotes to total number of packets generated when using the MCF protocol.

For the SRMCF protocol, under the same conditions, the total number of generated packets T_S (S stands for SRMCF protocol) is:

$$T_S = h\sqrt{N}\lambda_n N + \frac{\lambda_n}{m}\sqrt{N}. \tag{3.6}$$

From Eqs. (3.5) and (3.6) we have:

$$T_M = T_S + \frac{\lambda_n}{m}(N - h\sqrt{N}). \tag{3.7}$$

If the term $N - h\sqrt{N}$ is positive, Eq. (3.7) implies that the SRMCF protocol generates fewer packets than MCF. To prove this, we determine the maximum value of $h\sqrt{N}$. Figure 3.11 illustrates a line topology, which is the situation where the average hop is maximum. In this case:

$$\max(h\sqrt{N}) = \frac{\frac{N}{2}(N+1)}{N} = \frac{N+1}{2} \tag{3.8}$$

For $N > 1$, it follows that

$$\min(N - h\sqrt{N}) = N - \frac{N+1}{2} = \frac{N-1}{2} > 0 \tag{3.9}$$

Combining Eqs. (3.7) and (3.9) gives:

$$T_M > T_S \tag{3.10}$$

The inequality Eq. (3.10) shows that under the same conditions, using the SRMCF generates less traffic than using the MCF protocol. In other words, if r_M and r_S denote to the number of packets to handover from sender to receiver, then

$$r_M > r_S. \tag{3.11}$$

The new inequality (3.11) means that with the SRMCF protocol each communication requires fewer packets than with MCF. Therefore, the energy consumption is lower as well. From Eq. (3.3) and Eq. (3.11) it can be concluded that SRMCF is able to achieve a higher throughput than MCF.

3.3.2 Packet Header Length

As mentioned earlier (cf. Section 3.1.1.2), SRMCF uses variable-length headers for packets directed from BS to SNs. In addition, the packet header length changes along the path. Let h_{fix} represent the packet header length (in bytes) used in packets from a SN to the BS. The header length for packets in the other direction is

$$h_{size} = h_{fix} + \alpha H, \tag{3.12}$$

where α is the length of the MAC address and H is the number of hops between the current node and the destination. The value of H is decremented as the packet progresses along the path from BS to SN. In fact, the packet header length in different nodes is an arithmetic progression with common difference α. Considering the average path length ($h\sqrt{N}$) and the change of the packet header length, the average packet header size for SRMCF is:

$$\begin{aligned} h_{avg} &\approx h_{fix} + \alpha \left(\frac{\left(\frac{\lceil h\sqrt{N} \rceil}{2} \right) \left(1 + \lceil h\sqrt{N} \rceil \right)}{\lceil h\sqrt{N} \rceil} \right) \\ &\approx h_{fix} + \alpha \left(\frac{1 + \lceil h\sqrt{N} \rceil}{2} \right). \end{aligned} \tag{3.13}$$

For $h\sqrt{N} \gg 1$ the expression becomes

$$h_{avg} \approx h_{fix} + \frac{\alpha h\sqrt{N}}{2} \tag{3.14}$$

Equation (3.14) shows that, for networks with a large number of nodes, eliminating unnecessary path information for packets from the BS to a SN almost decreases the average header length by a factor of two.

3.3.3 Routing Table Size

The BS keeps a routing table with all optimum paths to reach the SNs. Obviously, the size of aforementioned table depends on the number of SNs and how they are distributed in the network. Because of the variable path in table, there is no fixed relation between the number of nodes and routing table size. To calculate the approximate size of the table, suppose that the average path size is ($h\sqrt{N}$). As mentioned in Section 3.1.2.2, each entry of the routing table includes fixed-size elements and a variable-length path. Therefore,

$$T_{size} \approx N \left(T_{fix} + \alpha(h\sqrt{N}) \right), \tag{3.15}$$

where T_{size} is the total routing table size in bytes, T_{fix} is the size of the fixed elements in an entry and α is the length of MAC address. For a large number of nodes the fixed part of the entry will be much smaller than the variable part. Then,

$$T_{fix} \ll \alpha(h\sqrt{N}) \Rightarrow T_{size} \approx \Theta(N\sqrt{N}). \tag{3.16}$$

Equation (3.16) indicates that the routing table size grows like $N\sqrt{N}$, a fact that should be considered in the design of the BS. The variable entry size increases the complexity of managing the table, which must support queries by node, periodic updates of the expiration counter associated with each entry, and node insertion and deletions. If the BS has enough memory to keep a large table, then using a table with a fixed entry size will be simpler and reduces processing time. In this case, the length of each entry has to be large enough to store the biggest path. If P_{max} is the maximum path size (in bytes), then

$$T_{size} = N(T_{fix} + P_{max}). \tag{3.17}$$

Although the BS usually has more resources than the SNs, it is still useful to compress the routing table. An approach to compression is based on the idea that, if an SN is recorded the in routing table, then its near-node is also in the table. Therefore, if a pointer that refers to the location of the near-node is used instead of the path information, the BS can generate the path by accumulating the path information. In this case the routing table size is

$$T_{size} = N(T_{fix} + i), \tag{3.18}$$

where i is the size of the index into the routing table. This method is able to significantly compress the routing table.

3.4 Simulation and Experimental Results

This section presents results from simulations of the SRMCF and MCF protocols and results obtained from an implementation using TelosB motes. The results of SRMCF for various scenarios are compared with the values obtained from simulations of the MCF protocol. For simulation, the protocols are implemented in OMNeT++ 4 with the MiXiM framework as explained in Section 3.2.2. The parameters for the SNs correspond to the specifications of the MICAz and TelosB sensor modules [Cro11, Cro13]. The experimental results are obtained from a small network with six TelosB motes shown in Fig. 3.12. In this case, the SRMCF and MCF protocols are implemented on the Contiki OS [Con12], an open-source operating system for tiny systems.

3.4.1 Simulation Parameters

The MICAz sensor is based on the ATmega128L microcontroller [Atm11] and a 2.4 GHz IEEE 802.15.4 compliant RF transceiver (CC2420) [Tex11]. TelosB is a similar mote with the same

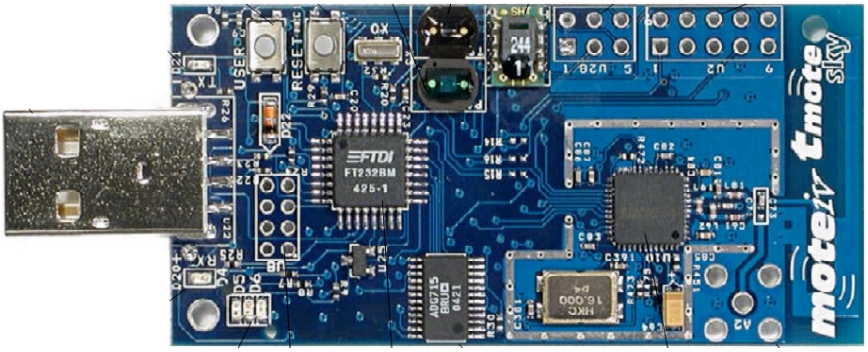


Figure 3.12: The photograph of the TelosB.

transceiver and a MSP430 microcontroller [Tex10]. The MiXiM framework includes all the code necessary to model a wireless environment including the physical layer, the radio channel, and the battery of the transceiver. Given the goal of low-power operation of the network, the Berkeley-MAC (B-MAC) protocol [PHC04] was used for the MAC layer, since it is a CSMA-based, energy-efficient MAC protocol.

Table 3.3 indicates the simulation parameters used for the wireless environment. Sensors are scattered randomly in a square area and where the BS node is located at the central point of the square. Table 3.4 lists the parameters used for the hardware implementation. In this case, we used contikiMAC, the energy-efficient MAC protocol implemented in the Contiki OS, which uses a wake-up mechanism with a set of timing constraints to enable motes to control the transceivers [KTDT12]. Although both B-MAC and ContikiMAC are energy-efficient MAC protocols, their use results in different performance, as shown in the following sections. To reduce the effects of electromagnetic interference, all experimental results were obtained in an anechoic chamber.

Figure 3.13 shows three different arrangements of SNs, which are used for evaluation and comparison of the protocols. A black circle indicates the BS and a gray circle a SN. The performance of a protocol, not only depends on the number of sensors, but also on the network topology, node scattering and density, and environment. *NW2* has a line topology, in which each node has at most two neighbors. In comparison with other topologies, a line arrangement has maximum hop count and all nodes involved in routing packets (except the last node). The node density in *NW3* is higher than in the other two networks. By increasing the node density, the number of nodes in radio coverage of each sensor increases.

3.4.2 Setup time

Figure 3.14 shows the setup time of the network as a function of the number of sensors for networks with up to 200 nodes. At the beginning of the simulation, each sensor starts to work at a random instant in the interval between 0 s and 1 s; the BS broadcasts the first *costADV* message in the same interval.

The setup time depends on the number of the sensors and increases from 3 s (10 sensors) to 9 s (200 sensors) for the SRMCF protocol with MICAz motes. The MCF protocol takes less time

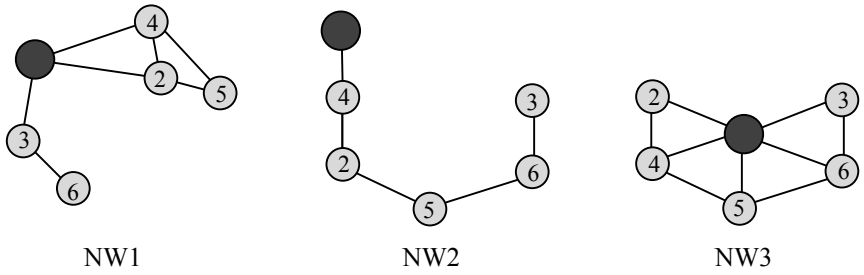


Figure 3.13: Networks with different arrangement of motes: NW1–random, NW2–series, NW3–BS in the center.

Table 3.3: Parameters for the simulation experiments

Parameter	**Value**
Mote	MICAz, TelosB
# Sensor nodes	50, 100, 200
Network area	300×300 m^2
Maximum packet length	256 byte
Antenna reach	25 m
Buffer size at SN	1024 byte
Simulated time	300 s
Data rate	250 kbps

Table 3.4: Parameters of prototype implementation

Parameter	**Value**
Mote	TelosB
Sensor nodes	6
Network area	10×10 m^2
Packet maximum length	128 byte
Antenna reach	1 m
Node buffer size	1024 byte
Simulation time	100 s
Data rate	250 kbps

Table 3.5: Experimental setup times for SRMCF and MCF protocols (in seconds)

Network	Time (s)			
	Simulation		Experimental	
	SRMCF	MCF	SRMCF	MCF
NW1	2.4	2.2	3.5	2.6
NW2	3.3	2.95	4.8	3.5
NW3	2.36	2.2	2.6	2.1

to complete the setup. The reason is that SRMCF setup is done in two steps (cf. Section 3.1.2), whereas the MCF protocol requires only the first. However, the SRMCF protocol requires setup to be performed only once and the difference is not high (less than 1 ss for 200 sensors). The SRMCF protocol does not need to execute periodic setup phases to ensure recovery from node failures.

MICAz motes complete the setup phase slower than TelosB motes. The ContikiMAC protocol generates more packets than B-MAC, a behavior which sometime allows the nodes to have a higher chance of receiving the messages, but also increases the number of collisions and packet losses. The SRMCF protocol with ContikiMac (on TelosB motes) takes considerably more time mainly due to the failure recovery mechanism. When a node does not get a *costADV* message, it broadcasts a cost request. Collisions may also occur with the MCF protocol, causing some SNs to not receive the cost message and to not turn on. Therefore, the setup time for MCF on TelosB shown in Fig. 3.14 is taken when the number of initialized SNs reaches 90 %, whereas the results for the SRMCF protocol on TelosB are for 100 % initialization. The same condition happens when the B-MAC protocol is used (MICAz motes), but the result is better, because in this case fewer collisions and lost packets occur.

Table 3.5 compares the simulation and experimental results for the setup time of the three prototype networks. Since the nodes in *NW2* must perform their setup in sequence, this network requires more time to complete the setup in both simulation and experimental conditions. As expected, the SRMCF protocol needs more time to complete the setup.

The quantitative differences between simulation and experimental results are mainly due to mismatches between the simulation environment and the actual behavior of motes. The antenna patterns and radio interference conditions are not exactly as simulated. The simulator assumes an omni-directional pattern, but the real motes exhibit different, more directional, patterns. Thus, depending on the mote orientation, the received signal can be better or worse than the simulated. The failure recovery mechanism in SRMCF also affects the experimental results.

3.4.3 Network Throughput and Packet Delivery

Figure 3.15 shows the total network throughput for both protocols with two different motes and as a function of the data rate generated by the BS. Each sensor generates packets at a rate of 2048

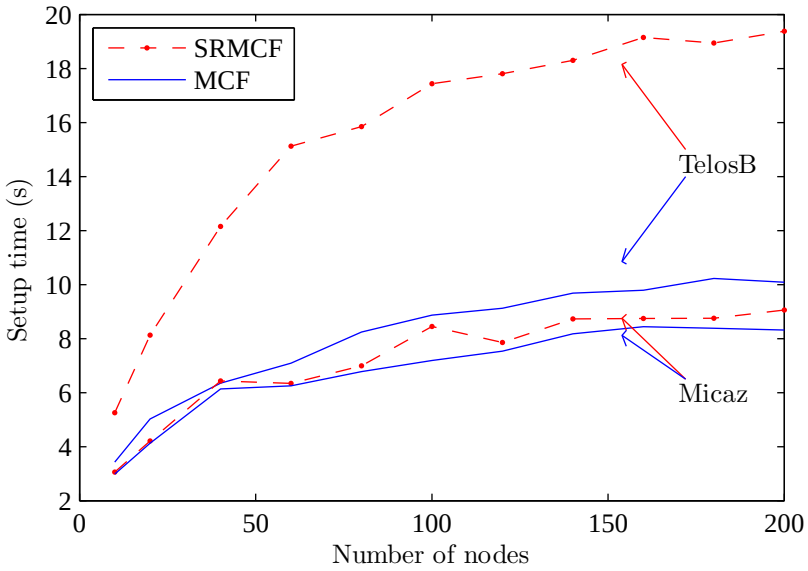


Figure 3.14: Network setup time for SRMCF and MCF. For SRMCF, all nodes are initialized within the setup time; for MCF, at least 90 % of the nodes are initialized.

bps. Networks with 50, 100 and 200 nodes were analyzed: for each size, 20 different randomly generated networks were simulated 20 times.

The relatively low throughput is mainly due to the MAC protocols. As Fig. 3.15 shows, MICAz with B-MAC has almost 3 times higher throughput than TelosB with ContikiMAC. This is caused by the higher number of packets generated by ContikiMAC, which increases the number of collisions in the network. However, the SRMCF protocol consistently achieves a higher throughput than the MCF protocol: 37.5 % for 50 nodes, 33 % for 100 nodes and 46 % for 200 nodes. The same happens with MICAz motes, however the improvements are smaller: 5.8 % for 50 nodes, 8.7 % for 100 nodes, and 8.6 % for 200 nodes. This is due to the fact that the SRMCF protocol is not subject to some of the problems associated with the flooding method used together with MCF, like implosion, overlapping and source blindness [ASSC02].

Figure 3.16 shows simulation and experimental results for packet delivery in the the prototype networks. In this figure, TX is total number of generated packets, RX,x(sim) represents the total number of received packets as obtained from simulation and RX,x(exp) is the total number of received packets as obtained experimentally. All the results are averages of 20 simulations or measurements, respectively. The BS and the SNs nodes generate packets periodically (5 packets/s). In total, 3000 packets are transmitted in 100 seconds.

The simulation results indicate that SRMCF protocol achieves 47%@*NW1*, 9.1%@*NW2* and 56.6%@*NW3* more packet delivery than MCF. For experimental results of packet deliver, SRMCF achieves 49%@*NW1*, 36%@*NW2* and 61%@*NW3* more than MCF. Both set of results confirm that the SRMCF protocol has better performance than MCF protocol, particularly when the node density increases (*NW3*). Because of less traffic with SRMCF, less collision happens and packet delivery increases.

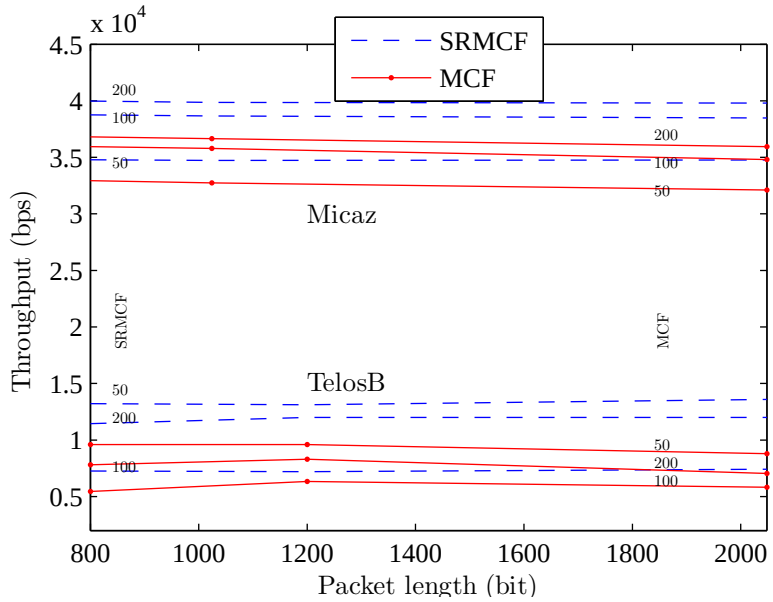


Figure 3.15: Throughput of the network in terms of the data rates of the packets generated by the BS (in bps)

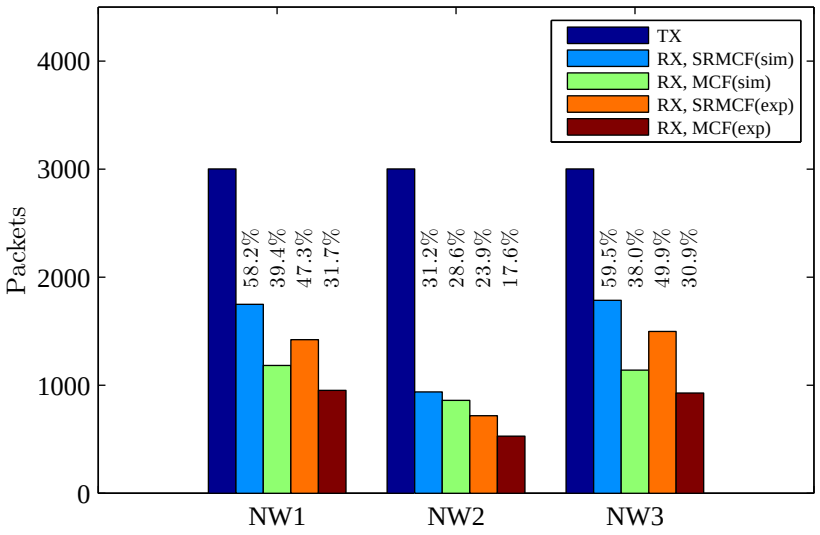


Figure 3.16: Simulation and experimental results for packet delivery with SRMCF and MCF protocols.

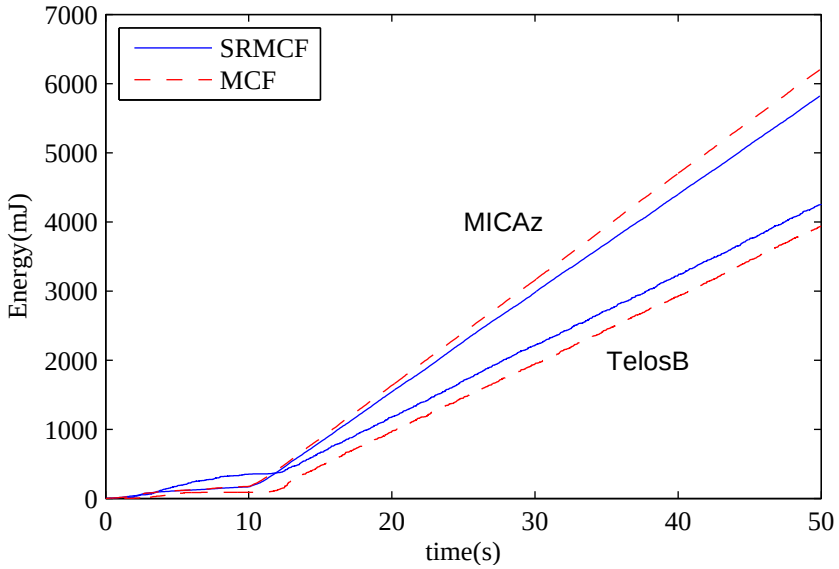


Figure 3.17: Energy consumption of network with 50 nodes (simulated).

3.4.4 Energy Consumption

Figure 3.17 shows the simulation results for the energy consumption of a wireless network with 50 nodes (for both MICAz and TelosB) when all nodes start to generate packets after $t = 10$ s. The amount of traffic generated by the SNs is the same as in the throughput simulations.

Under the stated conditions, energy consumption increases linearly with operation time in all cases (except during the setup phase). As can be observed, after 50 s the SRMCF protocol spends 6.7 % less energy with B-MAC than with ContikiMAC, due to fewer generated packets and to communication over minimum cost paths in both directions. As shown in Fig. 3.15, for a 50-node network the throughput for SRMCF with B-MAC is 27 % higher than the one for MCF. Therefore, compared to MCF, the SRMCF protocol consumes 26 % less energy for the same throughput. Since power is the time derivative of energy, the figures show that power consumption is constant after the setup phase. With SRMCF total power consumption is almost 141 mW, that is 6 % less than MCF (150 mW).

With ContikiMAC, the SRMCF protocol consumes 1.4 % more energy than MCF, but has 43 % better throughput. Therefore, in this case the SRMCF protocol consumes 29 % less energy than MCF for the same throughput. The total power consumption after setup with MCF is almost 96.2 mW, that is 1.4 % less than SRMCF with 97.6 mW. But as mentioned, MCF has lower throughput than SRMCF. In general, using ContikiMAC consumes less energy than using B-MAC, but also leads to lower throughput.

Lower energy consumption is expected to lead to longer network lifetime, as shown in Fig. 3.18. This figure characterizes network lifetime by showing the number of active nodes for the SRMCF and MCF protocols. To simulate the lifetime, sensors are supplied with limited amount of energy and the number of alive and responding sensors has counted. Each wireless sensor is assumed to

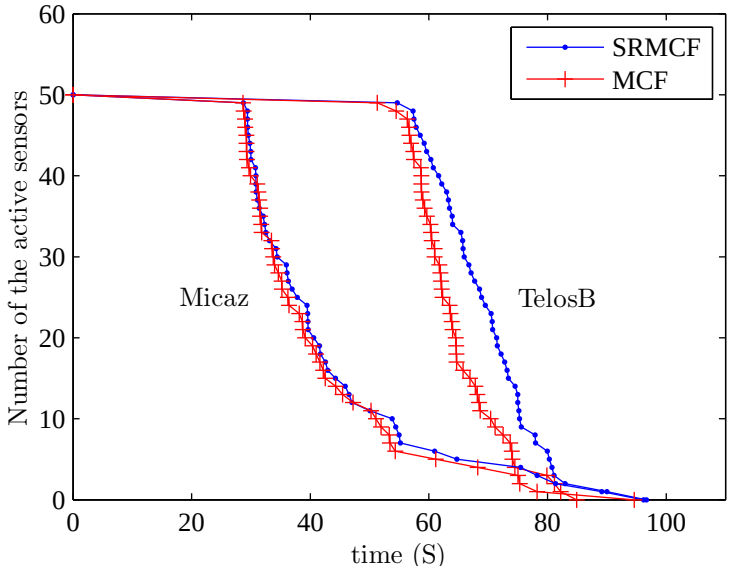


Figure 3.18: Number of the active nodes.

have a limited energy supply: a 3.3 V, 50 µAh battery for MICAz, and a capacity of 50 µAh for TelosB. The battery capacity for TelosB is reduced, because Fig. 3.17 indicates that this platform consumes less power than MICAz.

With MCF over B-MAC, after 36.2 s, 50 % of the nodes are still alive; with SRMCF, 50 % of the nodes are alive after 37.8 s, a 4.4 % increase in lifetime. For SRMCF with ContikiMAC, the lifetime increase is 10.6 %. As mentioned before, MCF generates more packets than SRMCF. Therefore, MCF with ContikiMAC, which also generates more packets than B-MAC, consumes more energy than SRMCF with ContikiMAC and has a shorter network lifetime.

Tables 3.6 and 3.7 show simulation and experimental results on the energy consumption of transmitters and receivers after running for 100 s, while each sensor generates a traffic of 640 bps. Energy consumption with SRMCF is smaller than with MCF for both transmission and reception. For transmission, SRMCF achieves 9.5 % *(NW1)*, 7.5 % *(NW2)* and 11.3 % *(NW3)* less energy consumption than MCF. In receiving mode the difference is more significant: 33.3 % *(NW1)*, 12.3 % *(NW2)* and 35.7 % *(NW3)*. These results indicate that the MCF receiver energy consumption is higher than the transmitter, specially when the node density is high. The experimental results of Table 3.7 confirm the simulation results.

3.4.5 Failure Recovery

As described in Section 3.1.3, failure recovery is based on monitoring near-node activity. If necessary, query messages are sent periodically with time interval T_q to inquiry about the status of the near-node. Small values of T_q lead to fast failure recovery, but require a large number of query messages, thereby increasing the power consumption and decreasing available throughput. With

Table 3.6: Total energy consumption (in mJ) after running for 100 s (simulation)

Network	SRMCF		MCF	
	TX	RX	TX	RX
NW1	403.56	403.38	417.24	620.01
NW2	389.88	404.87	428.87	445.21
NW3	350.21	395.91	434.34	627.48

large values of T_q, the generated number of query message decreases, but it takes more time to complete the recovery from the failure.

Fig. 3.19 presents the simulation results for energy consumption of the three prototype networks, from near-node querying as a function of T_q for 100 s of network activity. As it can be seen, the energy consumption decreases almost exponentially with increasing T_q. For $T_q < 4$ s, the energy consumption decreases rapidly as T_q increases; for $T_q > 4$ s it decreases at much lower rate

Network *NW3* displays the lowest energy consumption, because it has the highest node density of the three. Therefore, each node has more opportunities to receive a message from its near-node and most of the time does not need to send query messages.

Table 3.8 shows simulation and experimental results for failure recovery time in network *NW2* when there is no data traffic. For this study, *NNQR*=2 (cf. Section 3.1.2.2) and T_q=1 s or T_q=4 s. To simulate failures, nodes 4 or 5 are turned off for 20 s and then turned on. As expected, failure detection time for the case T_q=1 s is almost 4 times lower than for T_q=4 s. Recovery time is independent from the T_q value, because after failure detection the node runs in idle mode. The difference in recovery time is due to the different number of nodes affected by the failure: 5 nodes for a failure of node 4, and 3 nodes for a failure at node 5.

3.4.6 Routing Table and Packet Header Size

With SRMCF, packets generated by the BS have variable length, which depends on the number of nodes in the path between the BS and the destination nodes (cf. Sections 3.1.1.2 and 3.3.2). Both SRMCF and MCF have a fixed 5-byte header for packets generated by the sensor nodes. The implementation assumes that all sensors are from the same vendor. Therefore, the first three octets of MAC address, which identifies the organization that issued the address, are the same

Table 3.7: Total energy consumption (in mJ) after running for 100 s (experimental measurements)

Network	SRMCF		MCF	
	TX	RX	TX	RX
NW1	389.88	433.26	430.92	649.89
NW2	362.52	384.71	391.93	440.73
NW3	376.2	437	424.08	679.77

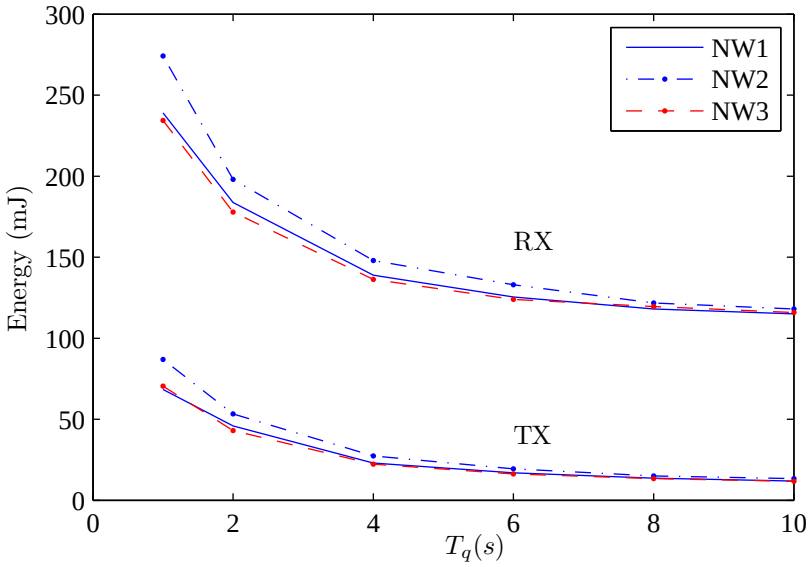


Figure 3.19: Energy consumption of query processing

Table 3.8: Failure recovery time in *NW2*

#	Event	T_q			
		1 s		**4 s**	
		sim.	**exp.**	**sim.**	**exp.**
N4	Detection (s)	2.8	3	12	9
	Recovery (s)	8.4	6	6	8
N5	Detection (s)	2	3	13.3	10
	Recovery (s)	4.3	7	4	6

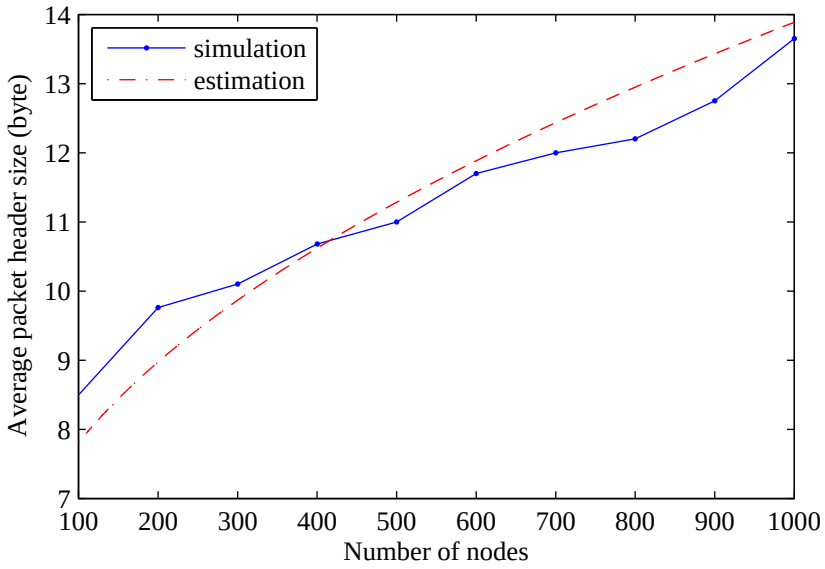


Figure 3.20: Average initial packet header size.

for all nodes. It is not necessary to include them in the routing path, making the packet header more compact (3 octets per each node address). Figure 3.20 shows the average packet header size obtained by simulating random networks with 100 to 1000 nodes. The figure also shows the calculated average header size given by Eq. (3.14). The results confirm that the average header size only increases by a factor of 1.6 when the number of nodes increases 10 times. Using a MAC address length $\alpha = 3$ and $h_{fix} = 5$, and using the simulation results for $N = 100$ to $N = 1000$, allows us to solve Eq. (3.14) for h, obtaining $h = 0.281$. The average header size is then:

$$h_{avg} \approx 5 + 0.281\sqrt{N}. \tag{3.19}$$

The dashed line in Fig. 3.20 shows that h_{avg} agrees with the simulation values.

As mentioned in Section 3.1.1.2, each node hands over the packets generated by BS to the next node with α bytes less in the packet header. With a fixed packet header (BS to node), the header size lies between 9.5 to 24.5 bytes (for networks with 100 to 1000 nodes). Therefore, by using the method described in Section 3.1.1.2, the SRMCF protocol achieves a significantly smaller average packet header size. It should be noted that both protocols have a relatively small header size in relation to the overall packet size (256 bytes) used in these simulations.

The average size of routing table for the uncompressed approach characterized by Eq. (3.15) was measured for randomly generated networks with 100 to 1000 nodes (20 samples for each network size). Figure 3.21 depicts those results together with the values estimated from Eqs. (3.15), (3.17) and (3.18). For this set of simulations, $T_{fix} = 5$ and the value of h obtained by interpolating the experimental results is 0.305. The simulation results agree closely with the calculated size given by Eq. (3.15). The simpler method of using only fixed-sized entries is acceptable for small or more compact networks, but the table size becomes very significant for larger ones: for 1000

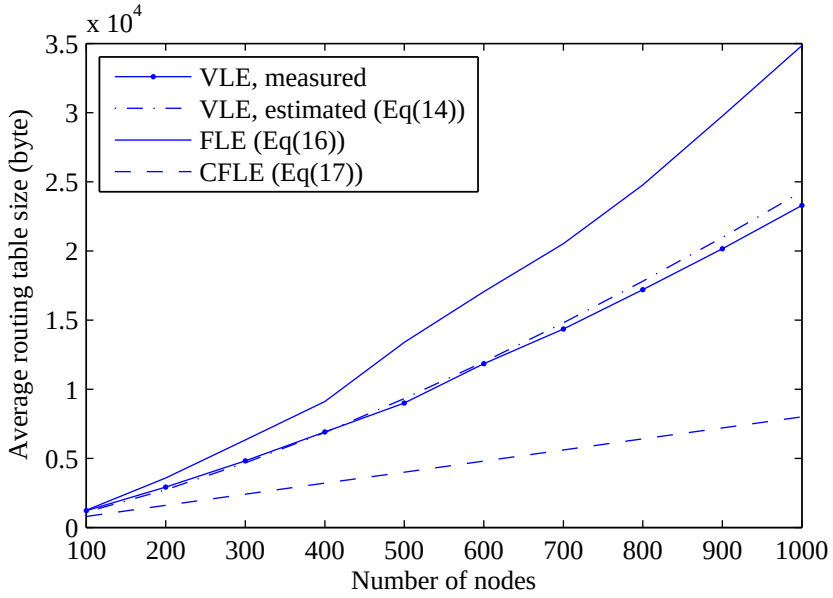


Figure 3.21: Average size of routing tables built using three different methods: VLE (variable-length entries), FLE (fixed-length entries), CFLE (compressed fixed-length entries).

nodes, the size calculated from the average maximum path length is 34.9 kB, which is 1.49 times larger than the approach that uses variable-length entries. The use of compressed tables would allow significant savings in memory at the cost of more complex search and insertion algorithms: the table size for a 1000 node network is predicted to be 8 kB by Eq. (3.18), a fixed value independent of network topology.

3.4.7 Throughput in Wired Networks

The SRMCF protocol will be used as a routing protocol for the wearable platform described in the following chapters. The performance of SRMCF will be evaluated further in the next chapter with a network of prototyped sensor nodes. In this section, the performance of both SRMCF and MCF protocols in a simulated wired network are evaluated and compared. To be more compatible with the conditions for SRMCF use in the next chapter, it is assumed that RTS/CTS handshaking at the MAC layer is used and that the node-to-node data rate is 10 Mbps. Each node periodically generates packets with 127 bytes resulting in an average traffic per node of 250 kbps.

Figure 3.22 shows the total throughput of randomly generated networks (10 networks for a given number of the nodes) as a function the number of the nodes. For small networks both the protocols have high throughput: 99 % for SRMCF and 98 % for MCF with 2 nodes. Throughput decreases as the number of the nodes increases, mainly with MCF. For networks with 20 nodes, SRMCF has 83 % throughput, but MCF achieves just 73 % throughput, which is 10 % is less than SRMCF. These results are similar to the behavior of the protocols in a wireless environment.

The power consumption of the communications module is mainly related to network activity,

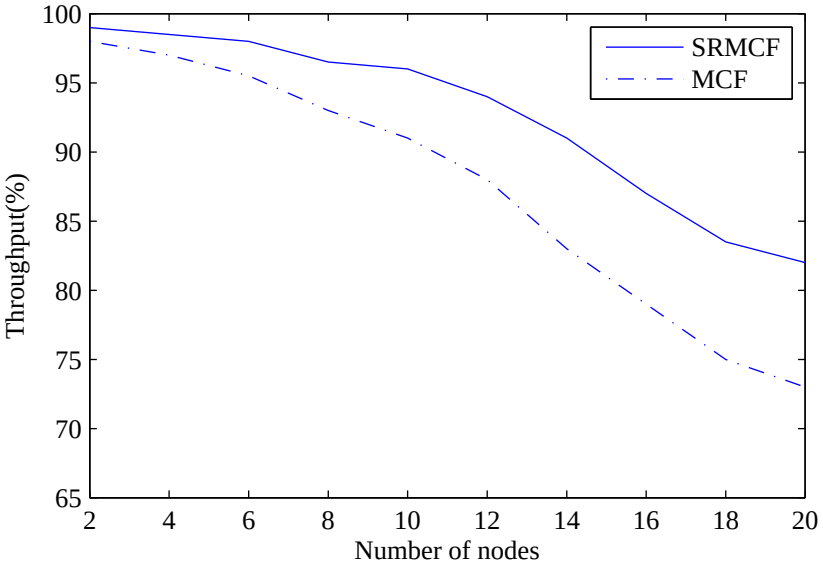


Figure 3.22: Total throughput in terms of the number of the nodes.

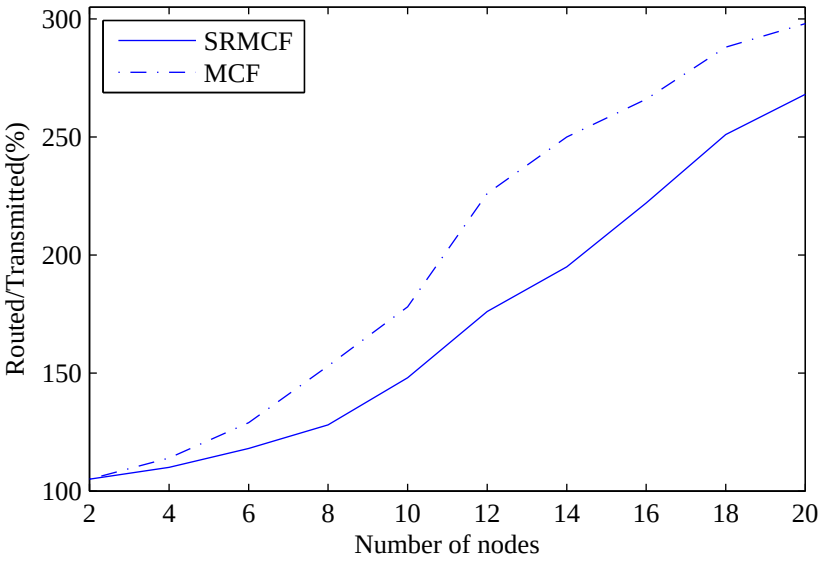


Figure 3.23: Ratio of routed packets to generated packets as a function of the number of nodes.

i.e., the number of generated and routed packets. In order to estimate the impact of the protocols on power consumption, the number of routed packet over generated packets was determined. Figure 3.23 depicts the results of the ratio of routed packets to generated packets obtained from the same networks as in the previous simulation. The results show that MCF needs to route more packets than SRMCF for the same number of data packets generated by the SNs. On the other hand, SRMCF has higher throughput. So, SRMCF consumes less power to handover the same number of packets.

3.5 Conclusion

The adoption of source-based routing in SRMCF leads to an improvement of performance over the MCF protocol. Unlike MCF, SRMCF applies the minimum cost forwarding method in both directions. The proposed protocol is energy-efficient, reactive and avoids routing tables at the sensor nodes. Only one routing table exists at the BS, which in general has enough resources to store and process the table. Absence of link and node failure control in MCF hampers its use in practical applications. The proposed failure recovery mechanism for SRMCF solves this limitation without affecting data communications. Additionally, the equal cost paths problem found in the MCF protocol does not occur in SRMCF. Not only has the proposed protocol higher throughput than MCF, it also dissipates less energy. The impact of the MAC layer was investigated through simulations with the ContikiMAC and B-MAC protocols. Simulation and experimental results indicate better performance of SRMCF when used together with ContikiMAC.

Based on experimental and simulation results, it can be concluded that minimum cost forwarding, failure recovery mechanism, absence of equal cost paths problem and improved performance make SRMCF a good candidate for energy-efficient, practical wireless sensor applications, as well as for wired. SRMCF is utilized as the routing protocol in the wearable system described in the next chapters.

Chapter 4

Wearable System Architecture

In this chapter the first prototype of the envisioned wearable system including network and hardware is described. The hardware of the system consists of SNs, BS, CPM and connections between the SNs, which are established by conductive yarns in a mesh network fashion. To the best of our knowledge, wired mesh networks for wearable systems have not yet been studied. So, to evaluate the behavior of the nodes and for network performance analysis, it is assumed that the network traffic can be modeled as a contention-based Poisson process with exponential arrival times in which nodes randomly generate traffic and end-to-end communication is based on packet switching. The experimental results obtained with this first prototype will be used to optimize system performance. First, an overview of the system is presented in the following sections. Then, the behavior of the sensor network, according to the characteristics of the system, is analyzed. The construction of a prototype of the system is subsequently explained. Experimental results obtained with actual circuits are also presented.

4.1 Design of the Network

Chapter 2 introduced BAN architectures for a variety of applications. Before discussing the intra-BAN network and the other parts, the architecture of the system will be introduced. The architecture of a BAN depends on the specific conditions of the application. In this section, the architecture of a wearable system that achieves the goals described in Section 1.3 is presented.

The system includes a wearable infrastructure and an on-body CPM which is in connection with a computer as shown in Fig. 4.1. The functionality of each part is as follows:

1. **Wearable infrastructure:** This part includes a set of SNs equipped with electromyography (EMG) electrodes for capturing electrical signals from skin tissue, and also accelerometers and gyroscopes for measuring kinetic parameters. To ensure a high data-rate and a reliable network, a mesh topology was selected. For that, SNs contain all necessary communication parts are connected to each other with conductive yarns. A BS node is responsible for collecting all data captured by the SNs. In fact, each SN acts as a router device to handover

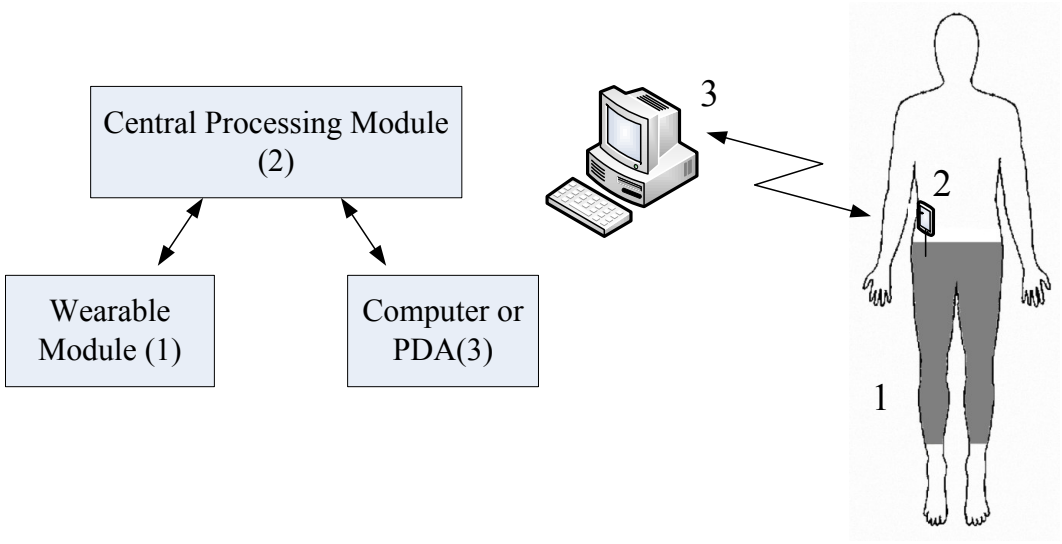


Figure 4.1: General architecture of the system.

packets from the source SN to the BS. The routing operation of the SNs is based on the SRMCF routing protocol already described in Chapter 3.

2. **CPM:** This part is a low-power microprocessor based system. The CPM, as an on body device, gathers the information from the wearable network via a communication link to the BS module. The central processing module accesses the different sensor nodes dispersed throughout the pantyhose using the wire mesh network embedded in the technical fabric. To transfer data to a computer or saving them, the CPM has been equipped with a wireless Bluetooth module, a USB port and a micro SD card.

3. **Computer or PDA:** The information collected by the CPM needs to be post-processed, analyzed and stored in a computer. A software program concurrently plots the real-time data transferred by CPM and shows the node status and connections.

4.1.1 Characterization of the Conductive Yarns

As referred in Chapter 1, the most comfortable and easiest way to monitor physiologic signals consists in using garments with conductive yarns as wire links [LG06a, GI07]. These kind of conductors can be modeled as an R-L series impedance [ZDD+12]. Because of the dependency of resistance and inductance on signal frequency and wire length, the impedance of a line of length l can be expressed as

$$Z(f,l) = l \times (R(f) + j2\pi fL(f)). \tag{4.1}$$

Table 4.1 shows the measured values of these parameters at 10 MHz for two different conductive yarns. One of the inherent properties of most conductive yarns is their ability to stretch: values are different for relaxed and stretched modes, and both resistivity and inductivity decrease with stretching. In comparison with normal copper wires, the equivalent inductive and resistive

Table 4.1: Measured values per unit [ZDD+12] (@10 MHz).

yarn	Relaxed (natural length)		Stretched (1.5 × natural length)	
	Ω/cm	nH/cm	Ω/cm	nH/cm
1	0.65	16.4	0.47	6.68
2	6.76	35	4.43	33

parameters are reasonably higher in yarns. For this reason, in practice several yarns must be put together in parallel to reduce the resistivity and inductivity to levels closer to those of copper wires. The number of parallel yarns needed is application dependent, and although the coupling between parallel yarns exist, it is lower than that found in a twisted pair made from copper wires (for the frequencies under consideration), so it was not taken into account in Eq. (4.1).

Supplying power to all parts of a wearable system is one important design consideration. Single battery operation, in alternative to having one battery per node, simplifies the apparatus. However, in order to enable power-supplying to sensors from one single central battery, the resistivity of the conductive yarns needs to be small to prevent losses.

4.1.2 Intra network

An intra-network of SNs in wearable systems includes a set of SNs in an interior network, all connected to a collector node. The topology of the network depends on the number of SNs and data rate. As mentioned in Chapters 1 and 2, the star and serial bus topologies are useful for a small number of the SNs. With increasing number, the performance of the aforementioned networks degrades: the number of ports increases in a star network and also the load on the bus in a serial network.

In networks with a high number of SNs the mesh topology is a good way to overcome these limitations while keeping high data-rates. To evaluate the channel behavior, consider a network with n SNs connected in series to a collector node as shown in Fig. 4.2. $C1$ to Cn stand for the capacitive impedance of SNs, and $R1$ to Rn and $L1$ to Ln for the conductive yarns equivalent circuit.

The combination of the electrical properties of the conductive yarn with the capacitance of the ports results in an Nth-order RLC low-pass filter for bus network and a second-order filter in a mesh network. Considering that the parameters of the conductive yarn depend on its length, the behavior of such a low-pass filter will also be a function of the yarn length. For sake of simplicity, the same length will be assumed for all of them. Under such assumption, the cut-off frequency (f_c) of the filter in both cases becomes [Che03]

$$fc = \frac{1}{2\pi\sqrt{L \times C}}. \tag{4.2}$$

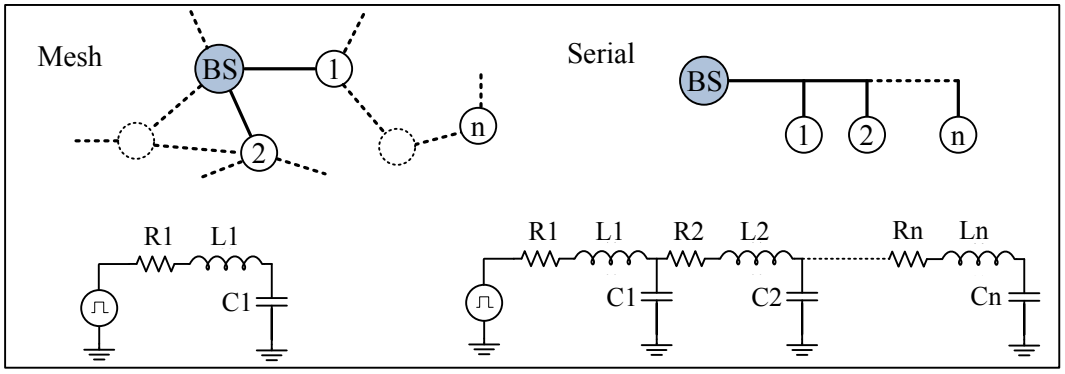


Figure 4.2: Serial bus and mesh Interconnection modes.

With increasing order of the filter, the signal attenuation increases and the -3dB cut-off frequency also decreases, as Eq. (4.2) shows. The phase of the signals will also be affected by the filter. The Fourier series of a square wave with levels between (0, A) and period T is

$$f(t) = A\left(\frac{1}{2} + \sum_{n=1}^{\infty} \frac{2}{n\pi} \sin\left(\frac{\pi n}{2}\right) \cos\left(\frac{2\pi n}{T} t\right)\right). \tag{4.3}$$

Therefore, the low pass behavior of the line generates a malformed signal at the receiver nodes, which is hard or impossible to detect because the attenuation and phase change the harmonics. Figure 4.3 shows simulation results for the frequency response of a serial network obtained with PSpice. The network consists of 16 nodes. In this figure, `v0` and `vm` represent the input and received signals, respectively, for a mesh network, and `v1` to `v16` denote received signals in a serial network. The interconnection lines are considered to be 50 cm long and made of typical conductive yarns (R=1.56 Ω, L=8 nH), and the input capacitance of the sensors is considered to be C=30 pF. It can be seen that with increasing number of nodes, signal attenuation increases significantly.

Figure 4.4 shows the simulation of -3dB cut-off frequency response in a mesh and in a serial bus network including 4 and 8 nodes for conductive yarns with lengths from 20 cm to 100 cm. The electrical parameters are the same as used in previous simulations. Both increasing the length of the yarns or the number of nodes reduces the available bandwidth.

Figure 4.5 depicts the -3dB cut-off frequency in terms of the node number for two different lengths of conductive yarns. For both, the frequency response decreases exponentially when the number of nodes increase.

According to the presented results, when the number of nodes in the wearable network increases, the network performance of a serial bus connection will degrade much more than the performance of a mesh network. Therefore, a mesh topology is a logical choice for high number of nodes and high data-rate.

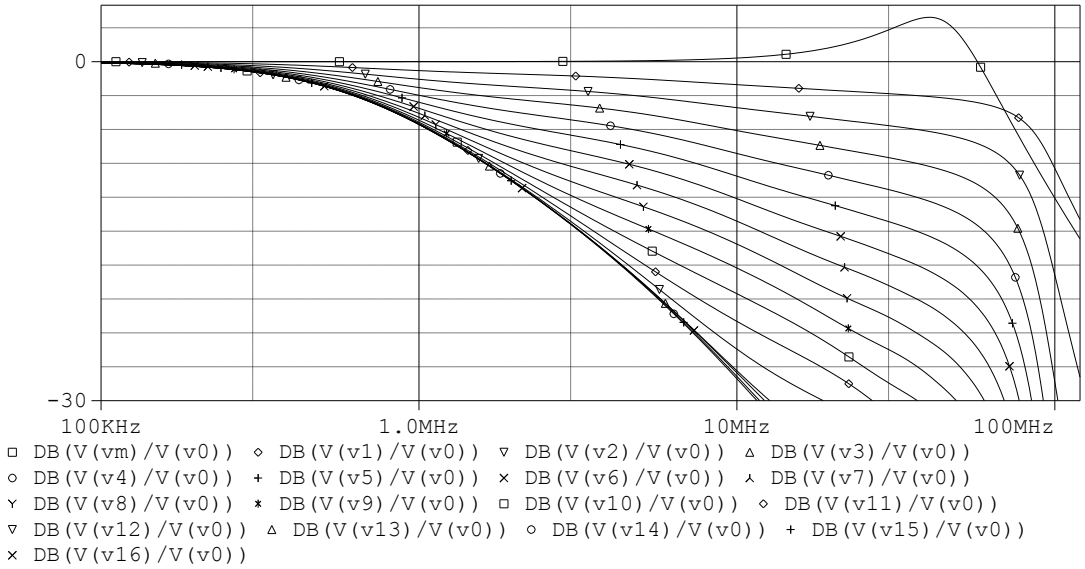


Figure 4.3: Frequency response of the serial bus network.

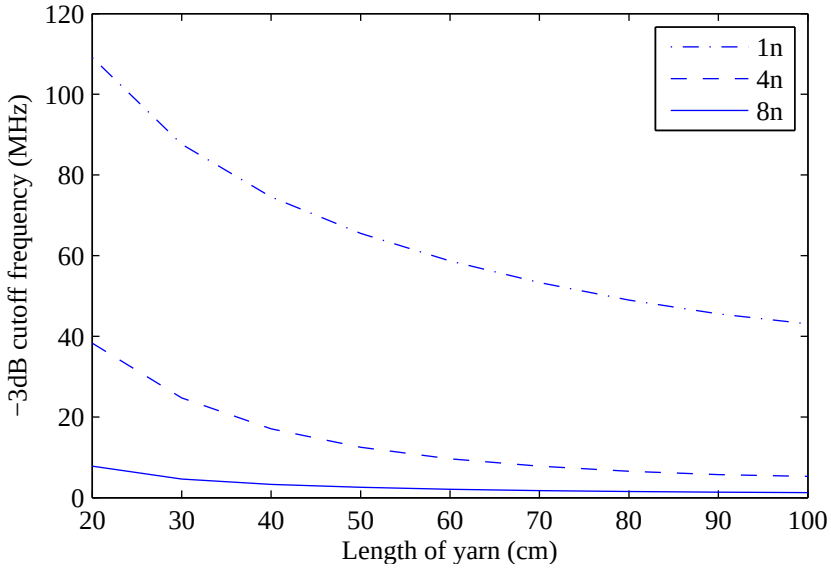


Figure 4.4: -3dB cut off frequency in terms of length of conductive yarns for different number of the nodes.

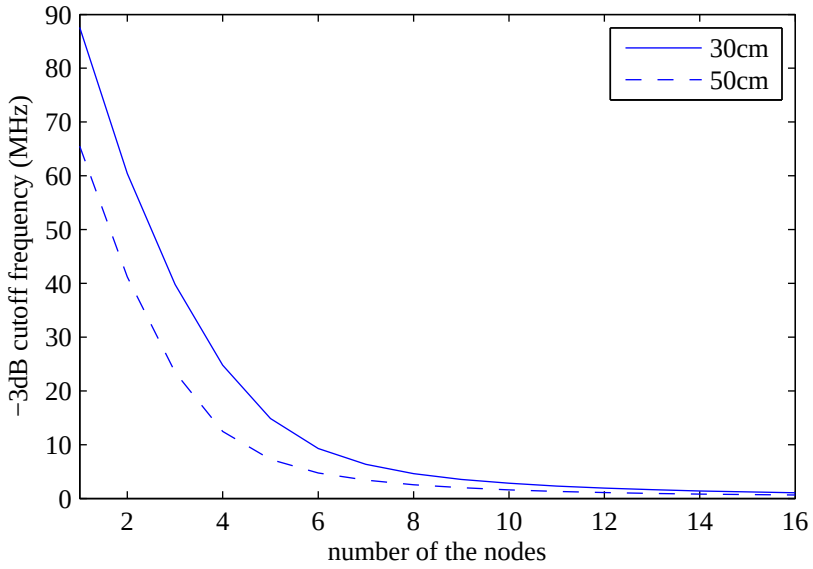


Figure 4.5: -3dB cut-off frequency in terms of node number for two different length of conductive yarn.

4.1.3 Hop-Count Bounds and Number of Ports per Node

In any multi-hop communication, to reduce the power consumption, collision, and end-to-end delay, the number of intermediate nodes has to be small, leading to a compact network. On the other hand, conductive yarns are vulnerable to mechanical stress, so in order to have a reliable communication, each node has to have several links available to use in case of link failure. In this subsection, the limits of network density and the number of sensor ports are evaluated.

In general, a set of sensors is able to form an arbitrary network. The hop count, which is the number of links between a node and the BS, depends on the node arrangement. For each given network, there is a maximum hop count. On the other hand, for a specific number of nodes, the maximum hop count is upper and lower bounded. The upper bound occurs when all nodes are arranged in a line with the BS at one end: the node at the other end has the maximum hop count, which is equal to the number of nodes.

In a compact network each node has many connections with others. To calculate the lower bound or minimum hop count (H_{min}) in the most compact network, consider a network composed of N sensors and a sink node BS as shown in Fig. 4.6. Suppose that the BS is at the center, each node on one side is connected to a node in a lower layer, and all the other ports of that node are connected to sensors in the upper layer. Here, the notion of layer is introduced to classify nodes based on their distance to the BS. Nodes in lower layers are nearer the BS than nodes in upper layers. Connections with uniformly distributed nodes on the branches make a full tree network of sensors with minimum hop number. Assume that all sensors and also the BS have K ports. So, the number of the sensors connected to the BS in the first layer is K. In the second layer, the number of the nodes is $K(K-1)$, because there are $(K-1)$ remaining ports for each node in the first layer.

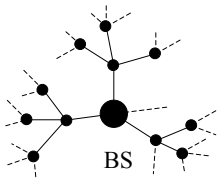


Figure 4.6: A random generated network.

By continuing this process, it can be deduced that

$$\begin{aligned} N &= n + K + K(K-1) + K(K-1)^2 + \ldots + K(K-1)^{(H_{min}-2)} \\ &= n + K\left(1 + \left(\sum_{h=2}^{H_{min}-1} (K-1)^{h-1}\right)\right), \end{aligned} \tag{4.4}$$

where n is the number of nodes in the last layer and must be in the range $\left(0 < n \leq K(K-1)^{(H_{min}-1)}\right)$, which is the minimum and maximum number of nodes in the last layer. So N is

$$N = n + K\left(\frac{1-(K-1)^{(H_{min}-1)}}{2-K}\right) \tag{4.5}$$

The value of n is positive ($\in \mathbb{N}$), so the inequality

$$N - 1 \geq K\left(\frac{1-(K-1)^{(H_{min}-1)}}{2-K}\right) \tag{4.6}$$

is always true. Rearranging Eq. (4.6) and taking logarithms, H_{min} is

$$H_{min} \leq \left(1 + \log_{K-1} \frac{(N-1)(K-2)+K}{K}\right). \tag{4.7}$$

The value of H_{min} is the largest integer that satisfies the condition, so

$$H_{min} = \left\lfloor 1 + \log_{K-1} \frac{(N-1)(K-2)+K}{K} \right\rfloor. \tag{4.8}$$

For a 4-port device, Eq. (4.8) becomes

$$H_{min} = \left\lfloor 1 + \log_3 \frac{N+1}{2} \right\rfloor \tag{4.9}$$

Equation (4.8) applies when there are no redundant links between the nodes. Consider now that each node uses k_r extra links between the nodes to increase the reliability of the network. In this case, the number of connections to the upper layer is $(K-1-k_r)$ and

$$\begin{aligned} N &= n + K + K(K-1-k_r) + K(K-1-k_r)^2 + \ldots + K(K-1-k_r)^{(H_r-2)} \\ &= n + K\left(1 + \left(\sum_{h=2}^{H_r-1} (K-1-k_r)^{h-1}\right)\right). \end{aligned} \tag{4.10}$$

Repeating the previous calculations gives

$$H_r = \left\lfloor 1 + \log_{K-k_r-1} \frac{(N-1)(K-k_r-2)+K}{K} \right\rfloor \geqslant H_{min} \tag{4.11}$$

Increasing k_r leads to a looser (less compact) network. Another conclusion that can be taken from Eq. (4.11) is that the base of the logarithm must be higher than 1. In other words,

$$K - k_r - 1 > 1 \quad \Rightarrow \quad K > k_r + 2 \tag{4.12}$$

Therefore, to have at least one redundant path for each node while having minimum hop count (most compact network) K must be equal to or larger than 4. So, the SNs and BS built for this work have four ports.

4.2 Design Considerations for Each Network Layer

The communication functions are grouped in 5 layers as is shown in Fig. 4.7: physical, MAC, network, middleware and application layers. In this section, the design considerations for each layer are presented.

4.2.1 Physical Layer

The communication medium, transmitter, receiver, coding and modulation define the first layer of the OSI model, the physical layer. To avoid the use of separated lines for transmitting and receiving, and also to reduce the complexity of the yarn network that must be embedded in the fabric, the communication between SNs uses bidirectional communication as shown in Fig. 4.8. Because the node-to-node communication is established over dedicated lines, a baseband communication method has been selected. Data is mapped to signal levels by the NRZI encoding method. In the idle state, the line is free and both nodes monitor the line for data. In fact, in this case the line is in high-impedance mode and both nodes are able to seize the line to start transmission. To avoid noise effects and keep the zero signal level in the idle state, the lines are pulled down.

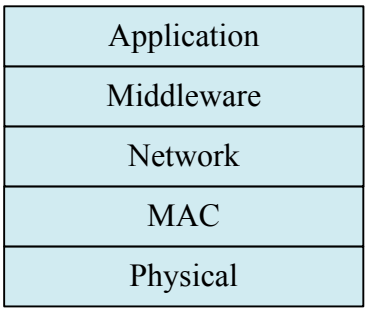


Figure 4.7: Network layers.

4.2.1.1 Baseband Signaling and Line Coding

This subsection presents the line coding employed in baseband communication over conductive yarns suitable for wearable systems. Transmitting data over the lines is a baseband communication and a line coding (also called digital baseband modulation or digital baseband transmission method) has to be used for data coding purposes. A line coding is a method for representing digital levels with analogue signals (amplitude and time). Selection of the line coding method depends on the specific properties of the application and channel. Over the years a variety of coding methods have been introduced. Among the existing binary coding methods, return-to-zero(RZ), non-return-to-zero (NRZ), non-return-to-zero-inverted (NRZI) and Manchester coding are probably the most widely used [Mad08, DR03]. The coding of an arbitrary string of data according to these coding methods is shown in Fig. 4.9.

Many communication, electronic and computer systems are based on unipolar NRZ. In unipolar NRZ a high voltage level stands for 1's and zero level for 0's. In NRZI a bit 1 is represented by a transition of the signal level and 0's have no transitions. In Manchester a 0 is expressed by a low-to-high transition, and a 1 by a high-to-low transition.

There are three main factors to take into consideration concerning the use of a coding method in a baseband communication [DR03, Wil95]:

1. **Baseline wander:** In systems including lines with Alternating Current (AC) coupling (either for power supply isolation or line matching), a *baseline wander* effect may occur, which may become specially critical if NRZ coding is used. Baseline wander results from Direct Current (DC) elimination and from the bandpass behaviour of the communication channel (line). The effect reveals itself as a fluctuation of the signal, which may lead to wrong decisions, thus increasing BER.

2. **Clock synchronization:** Successful data recovery in any kind of digital communication depends on how the receiver clock is synchronized with the incoming data stream transmitted by the sender. The utilization of global clock resources for the purpose is not, in general, an alternative for most distributed systems. Using a synchronization method then becomes inevitable. Under clock extraction for data recovery, somehow the spectrum information of the clock at the sender has to be sent to the receiver, possibly embedded in the data being transmitted. Coding plays here an important role. For example, in RZ or Manchester methods, for every bit of data, at least one transition occurs, which facilitates clock recovery on

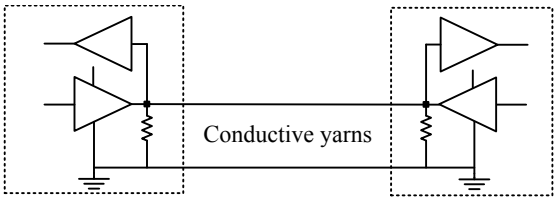


Figure 4.8: Connection between the nodes in physical layer.

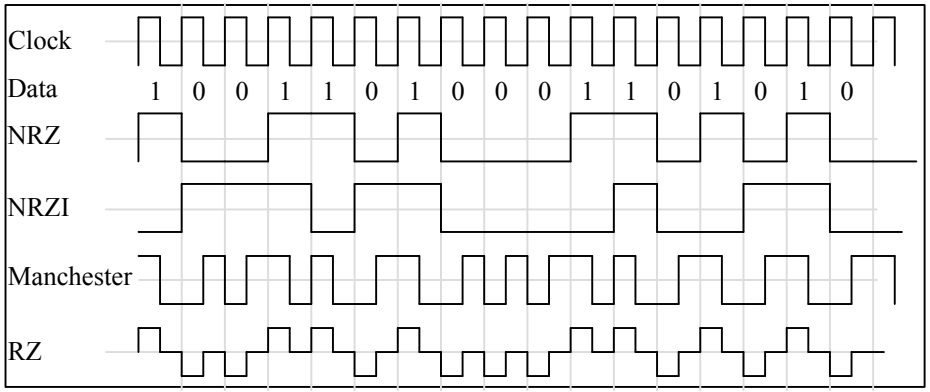


Figure 4.9: Coding a string of data with RZ, NRZ, NRZI, and Manchester methods.

the receiver side. In contrast, the absence of such a transition in NRZ or NRZI makes it difficult to perform clock recovery, specially for long string of 1's or 0's in NRZ and for long string of 0's in NRZI. In this case, a more complex clock recovery method is necessary.

3. **Line bandwidth:** All transmission lines are bandwidth limited. How the available bandwidth is used depends on the coding method. As mentioned above, RZ and Manchester coding embed the clock spectrum with the data. This extra information simplifies the recovery process, but at the expense of a lower bandwidth efficiency. In fact, compared to NRZ or NRZI, they need twice the bandwidth. In high speed systems with bandwidth limitations, coding methods like NRZ or NRZI should be used instead.

According to these considerations, in designing a system to carry high data-rate with conductive yarns either NRZ or NRZI should be selected. Unlike NRZ, which only works with DC coupling, NRZI can be operated in both DC or AC coupling because the information is uniquely defined by signal transitions. Such a property makes the design of the communication module more flexible. For example, if a Power Line Communication (PLC) mechanism is to be used with conductive yarns, for both communication and power supplying, then NRZI can be employed. For these reasons NRZI has been selected for line coding in this work.

Figure 4.10 depicts the signals at the receiver in the presence of white noise for NRZ (or NRZI with level decision). In this figure, k defines the decision level, corresponding to the average of the two signal levels. A_0 and A_1 denote the low and high level of the signal, representing the bit information: "0" and "1" respectively.

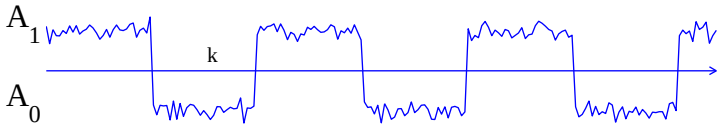


Figure 4.10: Received baseband signal affect by white noise.

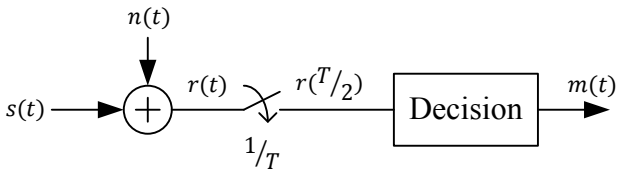


Figure 4.11: Sampling of received signal.

The signal detector is shown in Fig. 4.11. The receiver takes samples of the received signal at a rate of $\frac{1}{T}$, where T is the clock period that defines the data rate. The optimum sampling point is the middle of the incoming signal. After sampling and based on the k level, the *Decision* module detects the data. However, because of the noise that affects the transmitted signals, a wrong decision may be made during detection [CCR02, Mad08]. Figure 4.12 shows the Gaussian distribution of signals and their overlapping at point k:

$$k = \frac{A_0 + A_1}{2}. \tag{4.13}$$

The probability of error in the detection of data can be analyzed under the Additive white Gaussian noise (AWGN) model and is given by [CCR02]

$$P_e = P_0 P_{e0} + P_1 P_{e1}, \tag{4.14}$$

where P_0 (P_1) is the occurrence probability of 0's (1's) and P_{e0} (P_{e1}) is the probability of error detection of 0's (1's) as shown in Fig. 4.12. Since AWGN is a random variable with zero mean and variance σ^2, P_{e0} is given by

$$P_{e0} = \frac{1}{\sqrt{2\pi\sigma^2}} \int_k^\infty e^{-\frac{v^2}{2\sigma^2}} dv = \frac{1}{\sqrt{2\pi}} \int_{\frac{A_0+A_1}{2\sigma}}^\infty e^{-\frac{x^2}{2}} dx = Q\left(\frac{A_0 + A_1}{2\sigma}\right), \tag{4.15}$$

where Q represents the probability density function (pdf) of the error:

$$P(X < x) = Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-\frac{\gamma^2}{2}} d\gamma \tag{4.16}$$

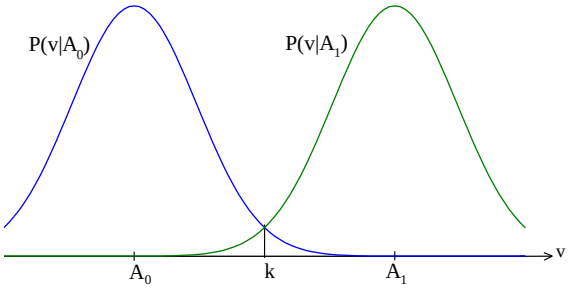


Figure 4.12: Probability distribution of signal levels.

In a similar way, for P_{e1} we have:

$$P_{e1} = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{k} e^{-\frac{v^2}{2\sigma^2}} dv \tag{4.17}$$

and

$$P_{e1} = P_{e0} = Q\left(\frac{A_0 + A_1}{2\sigma}\right). \tag{4.18}$$

The probability of error depends directly on the ratio of the received signal level to the AWGN at the receiver. For the system under consideration, as in general for BANs, the communication distance is in the range of few tens of centimetres. Although the electrical quality of conductive yarns is substantially worse than that of copper wires, the signal attenuation is small due to the short communication distances and the receiver's high input impedance. Simultaneously, noise in short-length wires has a very low level. As the measurements presented later in Fig. 5.17 show, these conditions are met by the present wearable network and the probability of error is in fact very small.

4.2.2 MAC layer

In mesh networking, nodes collaborate to propagate the data through the network. Each SN is able to communicate with its adjacent nodes. Sharing the communication medium implies, however, a MAC protocol in the data link layer to manage the access [IM05, YK12]. Selecting a MAC protocol depends on system requirements. As discussed in Chapter 2, in BAN applications, reliability, sharing of the line and energy efficiency are important features to take into account when devising a MAC protocol. To achieve the aforementioned features in the present system, communication at the MAC layer is controlled by the RTS/CTS handshaking mechanism presented in Fig. 4.13. In fact, in mesh networks it is very useful that each node is able to handle communication requests from different neighbors simultaneously. In order to efficiently manage the requests, the sender and receiver nodes should be aware of the status of each other. Otherwise, a significant number of data packets may be lost, requiring retransmission. The use of RTS/CTS handshaking helps to ensure reliable communication with low packet loss and good resource management.

The sender starts communication with an RTS message. If the receiver node is ready to receive the packet, it replies with a CTS message over the same line. Then the sender dispatches a packet encapsulated in a data frame. If the received packet is checked as error-free, the communication ends with an Acknowledgement (ACK) message generated by the receiver; otherwise an ERR message is sent to the peer node. After each message, the sender node waits for a reply from the receiver during a predefined time interval. The absence of reply within the specified time indicates a communication error. For the RTS/CTS handshake, if the sender node does not receive a CTS message in time, it repeats the RTS. If two nodes start to send RTS messages simultaneously, a collision will occur. To avoid the repetition of collisions, each node sends the a new RTS message after a random delay. It should be noted that all SNs are multi-task devices with 4 bidirectional

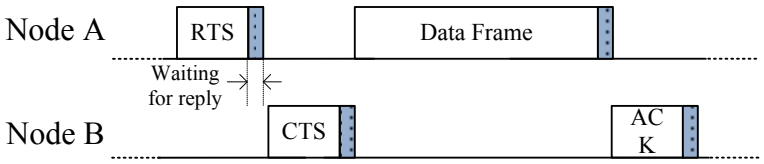


Figure 4.13: RTS/CTS MAC protocol.

ports. Therefore, RTS messages are not only used to share and control the line, but also to share the sensor resources, such as the packet buffer.

Figure 4.14 shows the frame used for MAC control and data frames. The shaded portion indicates that the line is free (high impedance). As can be seen in Fig. 4.14(a) and (b), the frames are formatted differently, depending on whether data or MAC control frames are being transmitted. Because the communication at the MAC layer is node-to-node over dedicated lines between the SNs, there is no need for explicit addressing support.

Control data reception depends on the synchronization of the receiver clock with the data stream. For that, start bits in the form of a string of 1's are used as a preamble signal (generates a pulse train). As described later, SNs support sleep and active modes for energy saving. If the receiver is in sleep mode, then the sender node has to somehow inform the receiver about the beginning of communication. Each SN is able to detect start bits even in sleep mode and then wake up. Therefore, start bits are not only used for synchronization but also for waking up the receiver node.

The receiver has to recognize the beginning of the data in the data frame. For that, *frame-sync* bits are placed just before the data stream. *frame-sync* is the 3 bit barker code "100". Barker codes are widely used in communication. A Barker code is a finite sequence of n values of +1 and -1 with the ideal autocorrelation property.

$$R_x(j) = \sum_{i=1}^{n-j} x_i.x_{i-j} = \begin{cases} n & j = 0 \\ 0 \; or \; \pm 1 & 0 < j < n \\ 0 & j \leq n \end{cases} \tag{4.19}$$

The output will be n only if the input sequence is exactly $\{x_1, x_2, ..., x_n\}$ (which is $\{1,0,0\}$ here). Otherwise it will be +1 or -1.

Fig. 4.14 shows that each data byte is followed by the bit "1". This is necessary because all nodes communicate in asynchronous mode, meaning that each node must extract the clock signal

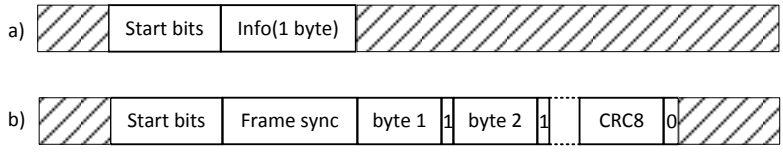


Figure 4.14: MAC frame: a) MAC control frame; b) MAC data frame.

from the incoming signals. However, because of NRZI coding, a stream of zeros is costly in this mode of operation. If the signal is kept constant for long periods of time, the receiver will not be able to recover the clock due to insufficient information (signal transitions). Therefore, a "1" bit is added after each byte of the data stream to ensure that at least one signal level transition occurs for every 8 bits of data. This synchronization bit is also used to define the end of a frame, by changing its value to 0.

Data frames are protected by a CRC-8 checksum [Max01]. Cyclic Redundancy Check (CRC) checksum is one of the most common and powerful error-detecting methods [Sta07]. Here the utilized code is the same as that used in the standard 1-Wire protocol [Max14]

$$G(x) = x^8 + x^5 + x^4 + 1 \tag{4.20}$$

4.2.2.1 Throughput at the MAC layer

Here, the throughput of the BS node for MAC layer communication is analyzed and the result is extended to the other nodes.

In packet switching or store-and-forward communication, delivering a packet from the sender to the destination node is done by node-to-node packet sending. Each node first stores the received packet, then forwards it to the next node on the path to the destination direction. The RTS/CTS handshaking mechanism is used to enable the sharing of the bidirectional lines and the receiver modules among the senders. Each SN is also a packet generator independent from the other nodes. Independent packet generation results, in the long run, in packets arriving at the intermediate nodes in a random sequence. Such a network operation can be considered as a Poisson process, with an exponential time interval between consecutive packets received at each receiver node. In a Poisson process, the probability of successfully receiving n packets in the period T is given by [YTT09]

$$P[n \text{ arrival in interval } T] = p_n(T) = \frac{(\lambda T)^n e^{-\lambda T}}{n!}. \tag{4.21}$$

Here, λ is the total packet rate, which is the sum of the packets generated by those nodes sharing a line. The cumulative density function (CDF) of the exponential distribution is given by

$$F_A(\tau) = P(A \leqslant \tau) = 1 - e^{-\lambda\tau}, \quad \tau \geqslant 0, \tag{4.22}$$

where A is the time interval between consecutive packet arrivals at the receiver. Then, the probability density function (PDF) will be

$$f_A(\tau) = \frac{dF_A(\tau)}{dt} = \lambda e^{-\lambda\tau}. \tag{4.23}$$

The expected value of A will be

$$E[A] = \int_0^\infty (1 - F_A(t))dt = \frac{1}{\lambda}. \tag{4.24}$$

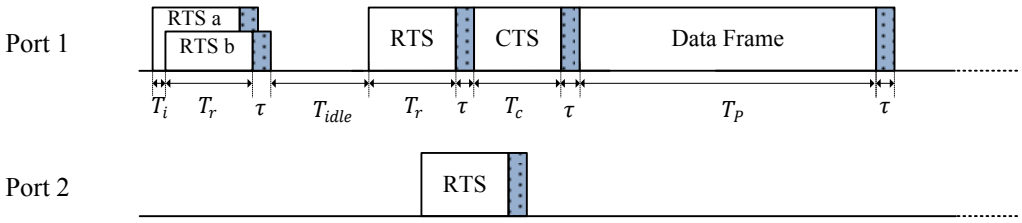


Figure 4.15: Messages at the ports of BS; *Port1* is connected to the SNa and *Port2* to the other SN.

In other words, if λ_d and λ_r denote the packet rate in the direct (from nodes to the BS) and reverse direction (from BS to the nodes) respectively, then

$$\lambda = \lambda_d + \lambda_r. \tag{4.25}$$

Since collisions in shared links depend on the traffic generated by each node, the separation of λ in two components is used to define throughput calculations at each of the nodes.

Suppose that the BS is receiving packets from a connected SN. Figure 4.15 depicts the signaling between BS and one of the neighboring nodes, let us say SNa. Here, T_i indicates the collision time, τ the propagation time including the signal propagation and detection time, T_r the RTS time, T_c the CTS time and T_P the total packet time. T_P also includes the ACK time, if used. Physical isolation of lines avoids collisions due to concurrent sending of neighboring nodes. Before starting communication, each node determines if the line is free, as in any CSMA/CA communication scheme. When SNa starts to send (after listening the signals over the line), BS will receive the signal after τ seconds. Due to the absence of information during this period (τ), BS may also attempt to send data or messages and will cause a collision. Hence, collisions may happen if $T_i \leqslant \tau$, and may occur between any pair of nodes. The time from the start of an RTS transmission by SNa to the end of the RTS issued by the BS represents the collision interval C given by

$$C = T_i + T_r + \tau \leqslant 2\tau + T_r \tag{4.26}$$

As mentioned earlier, the average idle time between packets is $\frac{1}{\lambda}$. The time taken to deliver a packet successfully includes all the messages exchanged, starting with RTS, and can be defined as

$$T_g = T_r + T_c + T_P + 3\tau. \tag{4.27}$$

In general, the throughput is the ratio of the time taken to successfully communicate, to the total time taken from the beginning of the communication attempt:

$$S = \frac{U}{I+B} = \frac{T_P P_U}{T_I P_I + T_B P_B} \tag{4.28}$$

S : throughput
U : average utilization period
I : average idle period
B : average busy period
T_P : average packet time
P_U : probability of successful arrival
T_I : average idle time
P_I : probability of idle time
T_B : average busy time
P_B : probability of busy time

Ideally, U, I and B are independent random variables. The busy portion, or B – the interval of time available for communication – has good (B_g) and bad (B_b) components. The good component refers to the busy period with P_g probability of successfully receiving the packets, while the bad component is a period of time with probability P_b of having no received packets.

$$B = P_g B_g + P_b B_b \tag{4.29}$$

A packet is successfully received if no other transmission starts within the collision period (T_i). So, from Eq. (4.21) the probability of successfully receiving the packet is

$$P_U = P[0 \text{ packet in } \tau] = e^{-\lambda_r \tau} = P_g. \tag{4.30}$$

A packet takes $T_P + \tau$ seconds if no collision occurs. In fact, this value corresponds to the good busy time (B_g). Considering Eq. (4.30), the probability of a bad busy period will be

$$P_b = 1 - P_g = 1 - e^{-\lambda_r \tau} \tag{4.31}$$

The idle time is the same expected value determined by Eq. (4.24)

$$I = T_I P_I = E[A] = \frac{1}{\lambda} = \frac{1}{\lambda_d + \lambda_r} \tag{4.32}$$

Now, using the calculated values, the throughput at the BS will be

$$S_{BS} = \frac{P e^{-\lambda_r \tau}}{\frac{1}{\lambda} + T_r + 2\tau + e^{-\lambda_r \tau}(T_c + P + 3\tau)} \tag{4.33}$$

The arrival rate λ is the time a packet takes while waiting in queue plus the packet serving time. Encapsulating the header of the packet at the receiver node for processing and routing purposes, also takes some time. When BS is connected to only one node, the idle time cannot be smaller than the packet serving time. Considering this time, the maximum channel utilization can be calculated:

$$S_{BS} = \frac{Pe^{-\lambda_r \tau}}{T_{idle} + T_r + 2\tau + e^{-\lambda_r \tau}(T_c + P + 3\tau)} \tag{4.34}$$

where T_{idle} is the idle time. For a BS connected to more than one node, λ is

$$\lambda = \lambda_r + \sum \lambda_i \tag{4.35}$$

For any other SN, the same conditions apply.

4.2.3 Network Layer

The routing protocol for the wearable wired network uses the same concepts explained in Chapter 3 for the generic SRMCF. The main difference is the addressing method since the communication medium is different. In A wireless environment, due to the use of shared channels, the communication between two nodes affects the other nodes located in the radio coverage area. Therefore, nodes have to use MAC addresses in order to distinguish sender and receiver nodes. There is no such problem in wired networks because nodes are interconnected with dedicated lines. In fact, the receiver recognizes the sender by the interface port number (which is always a number between 1 to 4 in the present implementation). Therefore, instead of using MAC addresses in the routing table of the BS, the port number is employed.

4.2.3.1 SN-to-BS Communication by Packet Switching

Here, the network performance in terms of number of packets stored in buffers, end-to-end delay due to buffering, and the probability of packet loss due to buffer overflow are analyzed using queuing theory. Since we are in the presence of a Poisson process with the exponential packet intervals, the model that fits is the M/M/1 in Kendall's notation [Geb08, BGdMT00]. Usually a queue is characterized in terms of packet arrival rate (λ), service rate (μ) and traffic intensity ($\rho = \lambda/\mu$). According to the Little's theorem, the average number of packets in a system (N) is given by

$$N = N_w + N_s = \lambda(T_w + T_s) = \lambda T, \tag{4.36}$$

where N_w denotes to the number of packets waiting in the queue, N_s the number of packets being served, T the average amount of time a packet spends in the system, T_w the time a packet waits in the queue, and T_s the service time. Eq. (4.36) can be extended to more complex networks. For a network including i nodes,

$$\sum N_i = T \sum \lambda_i. \tag{4.37}$$

For an M/M/1 system, it can be shown than the values of N and T are given by

$$N_w = N - \rho, \quad N_s = \rho, \quad N = \frac{\lambda}{\mu - \lambda} \tag{4.38}$$

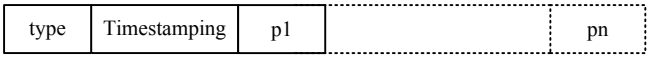


Figure 4.16: Payload format.

$$T_s = \frac{1}{\mu},\ T_w = \frac{\rho}{\mu(1-\rho)}, \quad \Rightarrow \quad T = \frac{1}{\mu - \lambda} \tag{4.39}$$

If the length of the generated packets is fixed then the service time will be fixed as well and the queue can be modeled as M/D/1. In this case, T_w will be

$$T_w = \frac{\rho}{2\mu(1-\rho)} \tag{4.40}$$

In systems with a buffer size B (M/M/1/B or M/D/1/B) the same equations are valid, but because of the limited size of the buffer, packet loss may occur. The packet loss probability is equal to the probability of the buffer being full or

$$P(B) = \frac{(1-\rho)\,\rho^B}{1-\rho^{1+B}} \tag{4.41}$$

Section 4.4.5 presents the simulation results of the behavior of the wearable infrastructure according to the aforementioned analysis.

4.2.4 Middleware and Application Layers

The middleware layer, placed between the network and application layers, is responsible for making the appropriate interface between the hardware, network, and applications, so that they operate as a whole. It provides services for driver applications and also provides a runtime environment that can support and coordinate multiple applications.

In the application layer, all acquired data will be saved to the microcontroller RAM. The middleware controls the length of data in the RAM and builds the payload as shown in Fig. 4.16. The first byte defines the payload type that it is used for addressing the payload contents. Time stamping includes two bytes that are used for synchronization purposes (cf. Chapter 5).

Depending on the application, a sensor can be used to monitor a phenomena, sensing, event detection and identification, etc. [IM05]. The application layer uses the underlying network layers to establish process-to-process communication. In the present system, each SN is able to capture EMG and kinetic signals by using three sensors: EMG electrode, accelerometer and gyroscope.

1. **EMG sensor:** It includes electrodes for capturing electrical signals from skin tissue. Raw EMG signals can be in the range $\pm$ 5 mV (for athletes!) and typically the frequency ranges between 6 to 500 Hz, showing most signal power between 20 Hz and 150 Hz. An Analog to Digital (ADC) with 12 bit resolution and 1 kHz or even 1.5 kHz sampling frequency is sufficient for most EMG applications [Kon05].

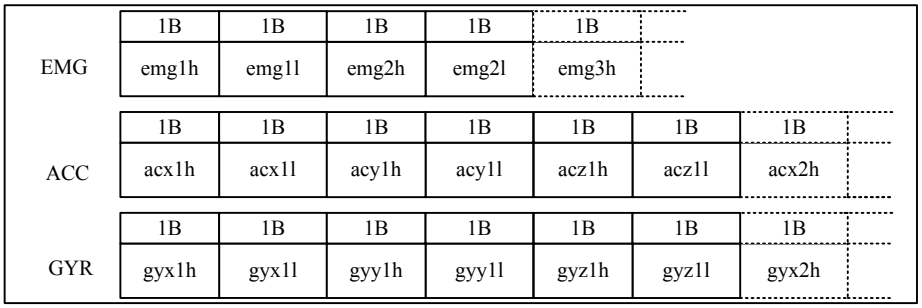


Figure 4.17: Data format in application layer.

2. **Accelerometer:** An accelerometer is a sensing element that measures acceleration and can be used to capture physical motion, which is a component that helps to categorize a person's activity [YWI+08]. Three-axis accelerometers fabricated with Micro Electro Mechanical Systems (MEMS) technology are widely used and have very small size.

3. **Gyroscope:** A gyroscope is a rotary rate measuring sensor [ZZ11]. These sensors are available in miniaturized sizes (MEMS technology) and use Coriolis effect to convert the rotary motion into a measurable linear motion. They are utilized for body motion detection.

The microcontroller utilized to develop the SN contains a multi-channel ADC; one of them, with 10-bit resolution, is used for EMG. The internal RAM is utilized to buffer all data acquired by the sensors. The data record format is as shown in Fig. 4.17. The ACC and GYR refer to the accelerometer and gyroscope respectively. Both sensors are of three-axes type, which means that each sample includes three values of 16-bit, sampled at 50 Hz. The EMG corresponds to a single 16-bit value and its sampling rate is set to 1 kHz.

4.3 First Prototype

This section describes the first SN prototype developed to evaluate the proposed communication infrastructure. But before the presentation of the actual functionality of the system modules, the next subsections introduce the major design characteristics.

4.3.1 Sensors, Base Station and Central Processing Module circuits

Figure 4.18 shows a block diagram of the SN. The circuit contains a 16-bit microcontroller and one FPGA, (low-power Actel AGLN125 [Mic13]) on the main Printed Circuit Board (PCB) board; the FPGA is used to implement the physical and MAC layers of the network. Although the FPGA is able to drive the lines directly, a separated board is used for protection and flexibility. Line drivers and sensors including electromyography, accelerometer and gyroscope, are on separate PCB boards. The electromyography sensor board was designed by Ruben Dias for the ProLimb project.

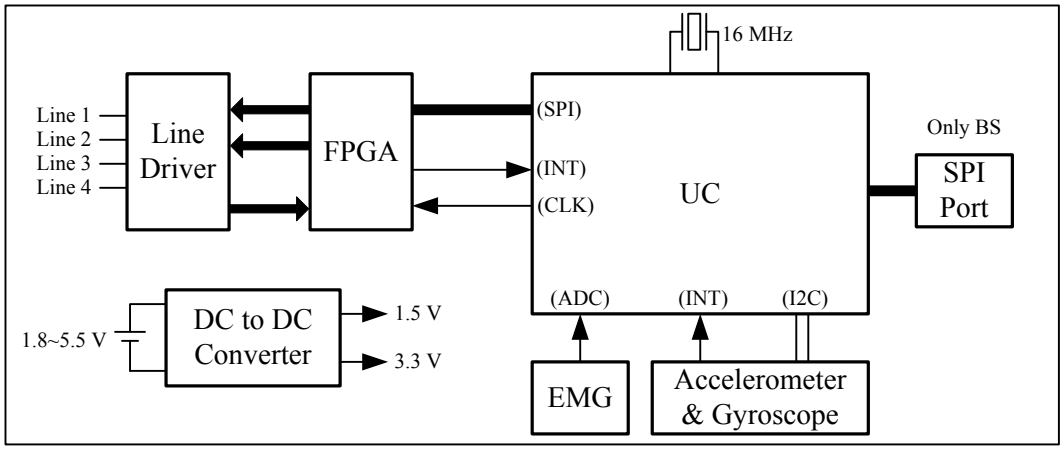


Figure 4.18: Overall organization of the SNs and BS.

Each SN has four bidirectional ports for connecting to other nodes in a mesh network. The power supply is a coin type battery (LIR2450), which together with a DC-to-DC converter, generates a 3.3 V power supply voltage for the microcontroller and sensors, and a 1.5 V supply voltage for the FPGA core and line driver circuits.

The internal oscillator of the microcontroller (16 MHz) is the clock source for both microcontroller and FPGA. The microcontroller implements the network, middleware and application layers, acquires the signals from EMG sensors and kinematic data from the accelerometer and gyroscope. The code was written in the C language with MPLAB X IDE and the microcontroller was programmed with the PICkit 3 programmer. The implemented circuit in the FPGA contains a number of control registers and all of them can be read or written by the microcontroller via an SPI port.

The BS circuit is in all similar to that developed for the SN, with the exception of an extra SPI port used to connect with CPM board. The CPM board (shown in Fig. 4.19) uses the same microcontroller and connects to the BS, and can perform three actions with the packets received from the BS:

1. **Send them to a computer via USB:** This is useful for realtime monitoring and debugging but prevents the user from moving freely.

2. **Send them to a computer via Bluetooth module:** It allows real-time monitoring while the user is in the range of the Bluetooth transmitter.

3. **Record data on a MicroSD card:** User data is recorded, but no real-time monitoring is performed.

4.3.2 FPGA-base Implementation of the Physical and MAC layers

The hardware implementation of the physical and MAC layers were described first in Verilog, synthesized with the Libero IDE Project Manager V9.1 from Microsemi [Mic12], and used to

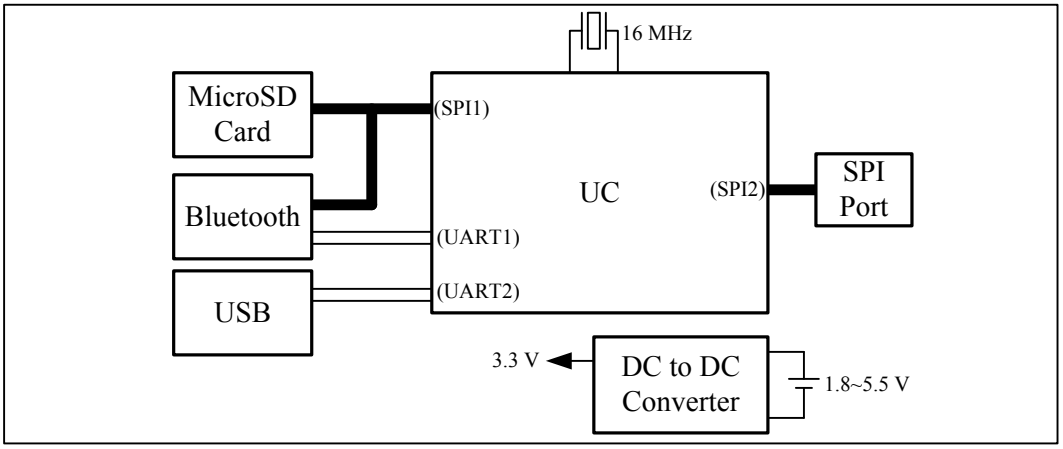


Figure 4.19: Block diagram of CPM.

program the FPGA with the FlashPro4 device programmer. Figure 4.20 depicts the block diagram of the communication module implemented in the IGLOO nano FPGA [Mic13]. The *Control* module provides access to the settings of the circuit and data reading from or writing to internal RAM via an internal bus and an SPI port. The SPI port provides a high-speed communication channel to the microcontroller. The acronym *TX* represents the transmitter module that includes all the sub-modules needed for communication: packet handover, encoding, generating and detecting MAC information, and also for frame generation. Conversely, the label *RX* indicates the receiver. This module also includes all sub-modules needed for decoding, generating and detecting MAC frames, buffering packets carried in data frames and for error checking. Both *RX* and *TX* modules are independent in order to enable communication with two nodes simultaneously. The *TX Line Switch* module is responsible for seizing the lines for transmitting data, either coming from the *TX* or *RX* modules. The *Signal Detector* module, as its name implies, detects the incoming preamble signals and RTS messages. The *Clock* module generates the internal clock, resets the circuit and sets the bit rate. All these modules and their functionalities are described in detail in the following subsections.

4.3.2.1 Internal bus architecture

The communication module includes two 8-bit internal buses; one for memory and buffer access by the microcontroller and the other for buffer sharing among *RX*, *TX* and microcontroller. Such a separation ensures the required high-speed operation.

4.3.2.2 Register bus

Circuit operation is completely controlled by a set of registers. The register bus, which is controlled by the *Control* module, provides direct access to all registers. But before moving on to more details, a more comprehensive discussion about the communication format between the microcontroller and the circuit is necessary for proper understanding of circuit operation.

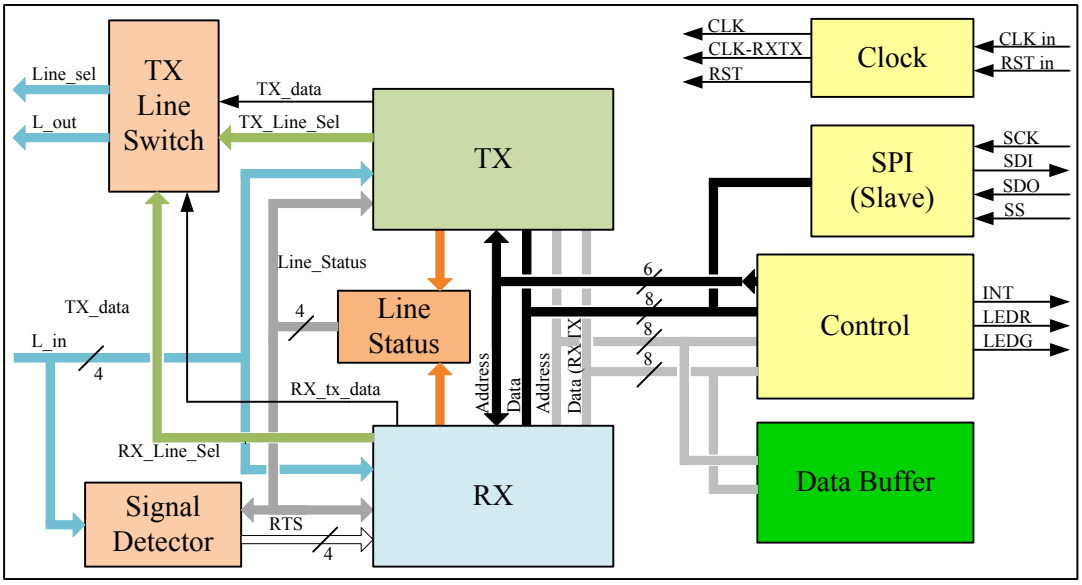


Figure 4.20: Communication module implemented on a FPGA.

First, the microcontroller uses the SPI interface to initiate one or a set of continuous instructions by changing the value of the Slave Select (SS) pin (which is configured as active-low pin). After activating SS, the microcontroller sends a one byte *command* (Fig. 4.22.b) to the control module (*B1* in Fig. 4.22(a)) and the *Control* module analyses it for the appropriate operation. The two *CMD* bits (7-6) determine whether the command is a reading or writing instruction. The rest of the command carries the register address. The two upper bits of the address (5-4) determine the location of the module that includes the register. In fact, in each module connected to the bus, an address decoder selects the module for reading or writing of the register. The remaining bits (3-0) are the register address inside the selected module. Thus, the maximum number of registers that each module can address is 16.

The bytes after *command* byte B1, namely *B2...Bn*, represent data to be written in a specific location or dummy data to be read. If only a single data byte is to be read or written, then the address in *B1* will be the register address, but for a consecutive series of registers it will represent the address of the first register. For example, if the microcontroller tries to write 3 bytes of data to the registers starting from 0x0A, then *B1* will contain 0x0A and the registers in 0x0B and 0x0C will be selected consecutively after the first register. For such a multi-register reading and writing,

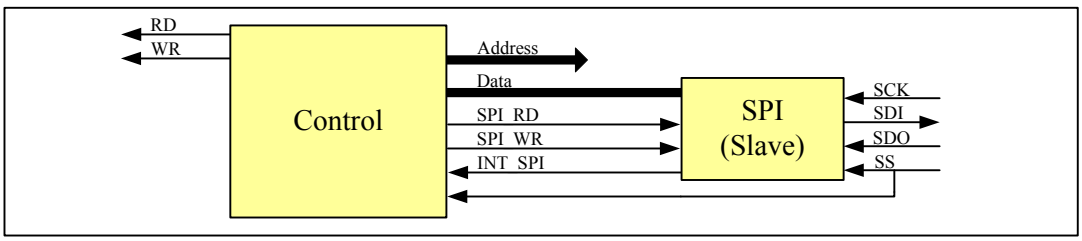


Figure 4.21: Interface for accessing the registers of the internal modules.

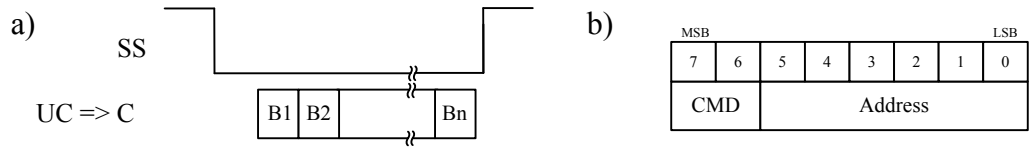


Figure 4.22: Communication format via SPI port.

a submodule in *Control* keeps the first address (0x0A) and generates the remaining by using an up-counter that is incremented together with the incoming data from SPI. It should be noted that, with 4 bits, the maximum addressing value will be 0x0F, but the counter stops counting whenever it reaches 0x0F to avoid overwriting unwanted registers. As mentioned before, the microcontroller has reading and writing access to the packets recorded in the internal RAM of the FPGA. For RAM addressing, the value 0x0F is selected, as explained later.

To improve the communication speed with the microcontroller, the *Control* unit gives the SPI module direct access to the registers for both writing and reading, as follows:

1. **Read:** Figure 4.23 depicts the timing diagram for reading values from a register. *INT-SPI* changes to high level at the end of each byte received by the SPI module. Then the *Control* unit reads the selected register by activating *RD* and writes its contents directly to the SPI with *SPI-WR*. Then the SPI sends it back to the microcontroller through the *SDI* pin. For the next register, the address increases by one during the reading process.

2. **Write:** The writing procedure follows the time sequence shown in Fig. 4.24. At the end of data reception by the SPI module, *INT-SPI* changes to high. Then the data values are stored in the target register by concurrently reading the SPI buffer (by activating *SPI-RD*) and writing (by activating *WR*) directly to the register.

4.3.2.3 Data bus

The internal RAM of the FPGA is used as a shared memory for the *TX*, *RX*, and the microcontroller. These three modules operate independently from each other to ensure simultaneous data reception and transmission. The submodule *Data Buffer Control* depicted in Fig. 4.26 provides the sharing mechanism and allows other modules to access the RAM via *Data bus*. The timing diagram of memory sharing for both reading and writing operations is portrayed in Fig. 4.25.

As mentioned earlier, each packet can contains up to 256 bytes. So, the memory is segmented to store one packet in each segment and allows each module to access a different segment at the same time (eight segments for a 2 kB RAM). Any reading and writing starts with a request, e.g. the *RX-WR-Req* for writing data received by the *RX*. The *Data Buffer Control* unit informs the *RX* by *RX-EN* if the access for writing is allowed, and puts other requests, if any, in a queue. Then the *RX* writes the data and releases the bus.

Figure 4.27 shows the format of a buffer segment. The three first bytes are reserved for meta-data control and information type. The information refers to the status of the segment, packet

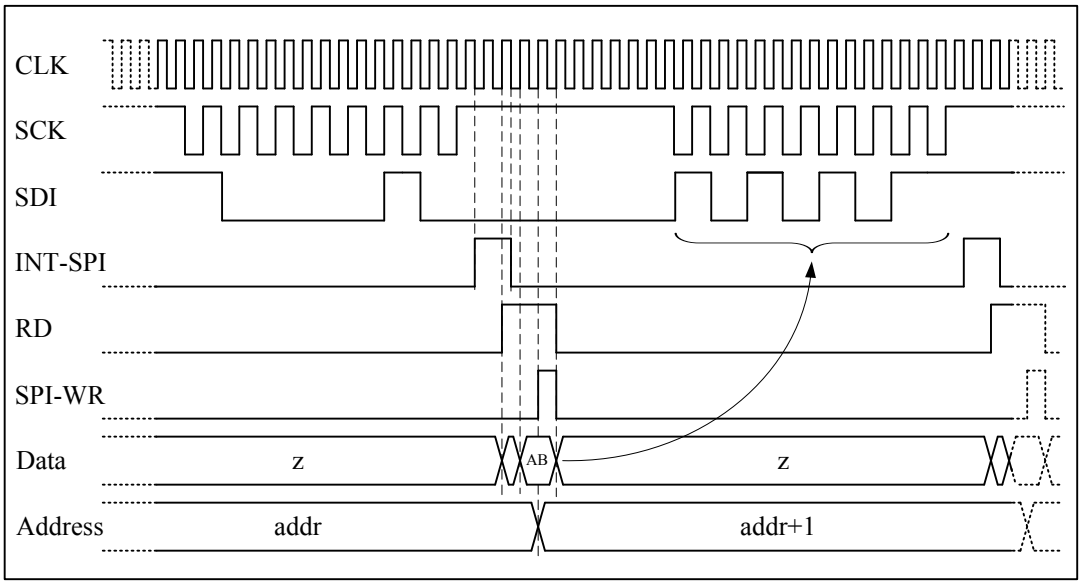


Figure 4.23: Reading a register value and sending it directly to the microcontroller.

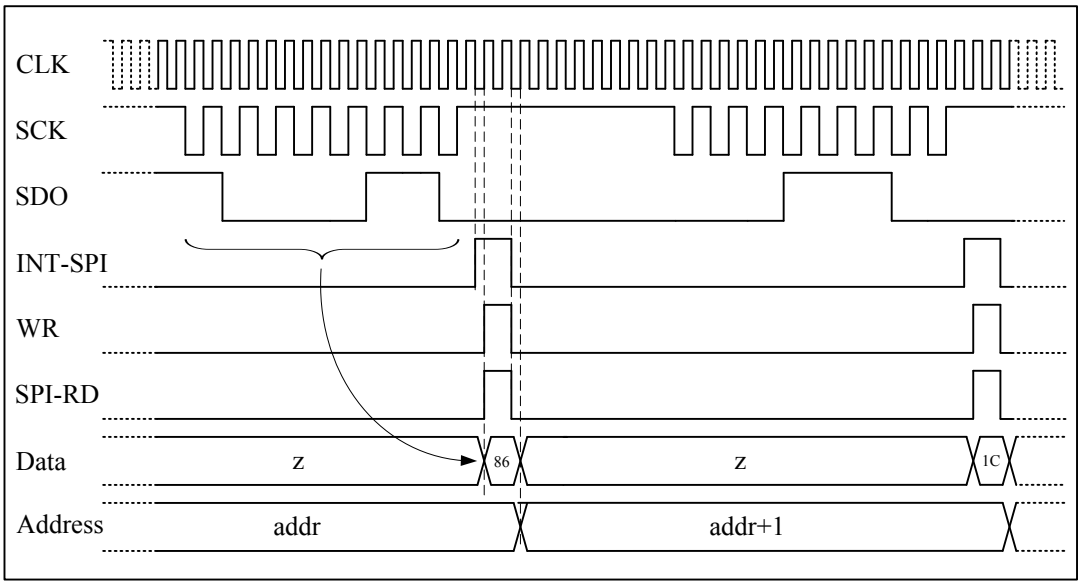


Figure 4.24: Microcontroller writes a value to a register in the communications module.

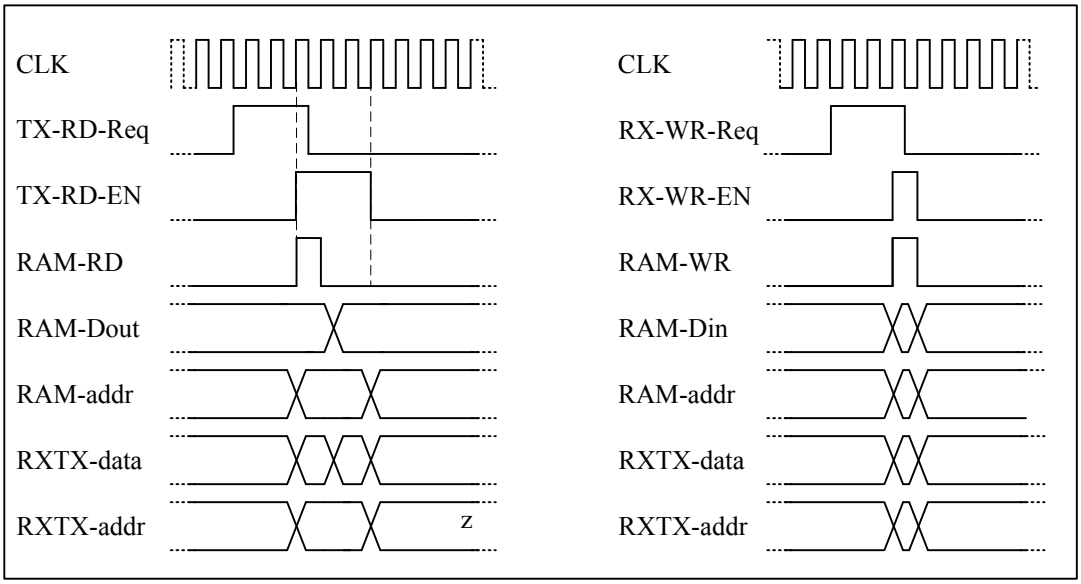


Figure 4.25: Timing diagram of reading and writing in data bus.

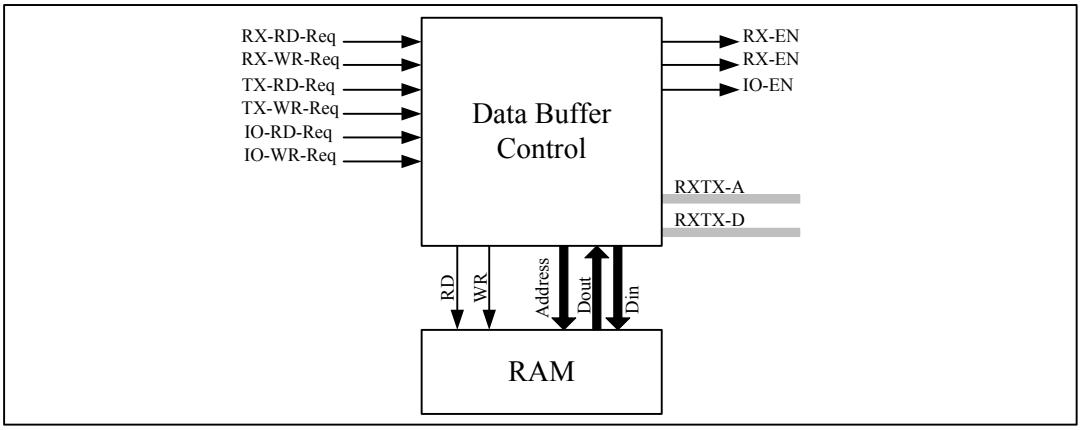


Figure 4.26: Block diagram of *Data Buffer* module.

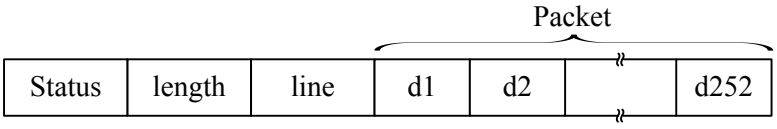


Figure 4.27: Buffer segment format.

length, and destination port number. Metadata limits the packet space to a maximum of 253 bytes. A submodule in *Data Buffer Control* checks the *Status* of each segment after being released by the user module. For example, if *RX* completes a packet reception, it will change the *Status* to *packet received* mode and releases that segment. The *Data Buffer* in its turn checks the *Status* and informs the microcontroller about the received packet.

4.3.2.4 Signal Detector

This module includes four *RTS Detector* submodules as Fig. 4.28 shows. Each *RTS Detector* is connected to one of the incoming lines and is able to detect RTS messages and inform the *RX* module to proceed with packet reception. In fact, the RTS message is a string of 1's with at least four consecutive 1's. Each module counts the number of consecutive 1's and activates the output. Any string of consecutive 1's that is smaller than four is considered as an error detection when the port is in idle mode.

4.3.2.5 Transmitter Module

Figure 4.29 shows all the submodules that implement packet processing at the transmitter. *TX Control* contains all the registers for transmission. It is responsible for controlling all transmission steps and holds the necessary submodules to realize the communication at the MAC level. The *TX Timing* circuit module generates the timing clock *CLKTX*, which determines the transmission data-rate. Its frequency is one quarter of the system clock *CLK*. The *Encoder* unit includes all submodules for data encoding and insertion of the synchronization bits in the data frame, as shown in Fig. 4.14(b). It generates the RTS messages and determines the end of transmission. The input of the *Encoder* is serial, supplied by a *PISO* (parallel-in serial-out) module that serializes the buffer output. An NRZI encoder module converts the NRZ output of the *Encoder*.

Transmission starts when the packet stored in the buffer is ready to be sent. As mentioned before, *Buffer control* controls the status of the packets (first data set of each segment in the buffer). If the status indicates that a packet is ready for transmission, then *Buffer controls* informs

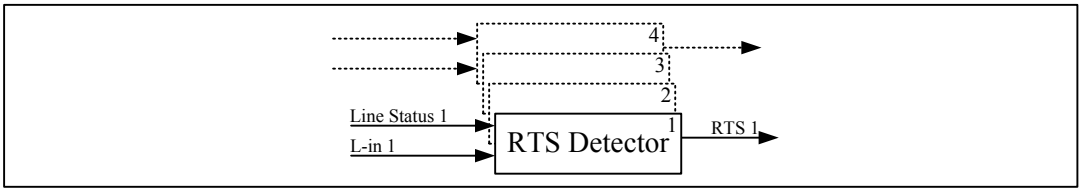


Figure 4.28: Signal detector module.

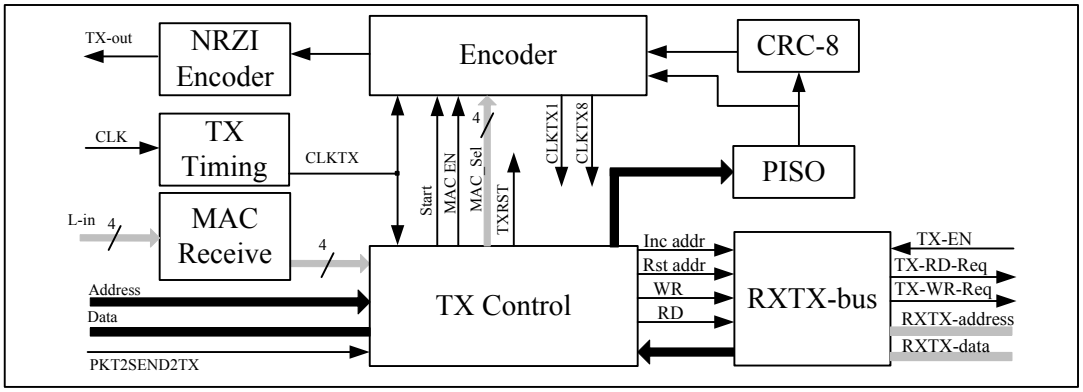


Figure 4.29: Block diagram of the transmitter module.

the *TX* by activating *PKT2SEND2TX*. Then *TX* starts the transmission process by first reading the packet information, including the length and the port number (1 to 4). Then *TX* checks if the line is available by using the *Line-Status* signals generated by the *Line Status* module. If the line is free, then *TX Control* activates *MAC-EN* for generating the RTS message through the *Encoder* module. After sending RTS, *TX Control* waits for a CTS reply from the receiver node. By utilizing the *MAC Receive* submodule, the *TX* module has the ability to receive MAC messages directly and independently from the *RX*.

The absence of a CTS reply means that for some reason the receiver is not able to receive a packet at that moment. So, the transmitter waits for a while and attempts another transmission. The transmitter starts to send the packet after receiving CTS. A counter in *TX Control* counts the number of outgoing data bytes and compares it with the packet length. While sending the packet, the *CRC-8* module generates a checksum of all the bits in the packet for error detection; the generated CRC value is added by the *Encoder* to the end of the packet before sending. If the destination node receives the packet correctly (no CRC error), then it replies to the sender with an ACK message. Upon completion of this cycle, the *TX* releases the packet in the buffer by changing its status to free. The absence of an ACK response or the reception of an *ERR* message means that an unsuccessful transmission took place and the *TX* needs to resend the packet. The number packet transmission retries time is determined by the *Repeat RTS* register in *TX Control*.

4.3.2.6 Receiver Module

All submodules of the *RX* module work together to implement the packet reception process according to the block diagram of Fig. 4.30. *RX Control* is responsible for controlling all steps in packet reception. Like *TX*, *RX* is an independent module capable of generating and sending MAC messages and communicating directly with the sender. *RX timing* generates a clock signal synchronized with the incoming stream for data recovery and also detects the frame starting bits for alignment. The *Decoder* decodes received data and also generates the *CLKRX1* and *CLKRX8* signals. These are used by the *SIPO* and *CRC-8 check* modules for byte formation and separation of synchronization bits from data.

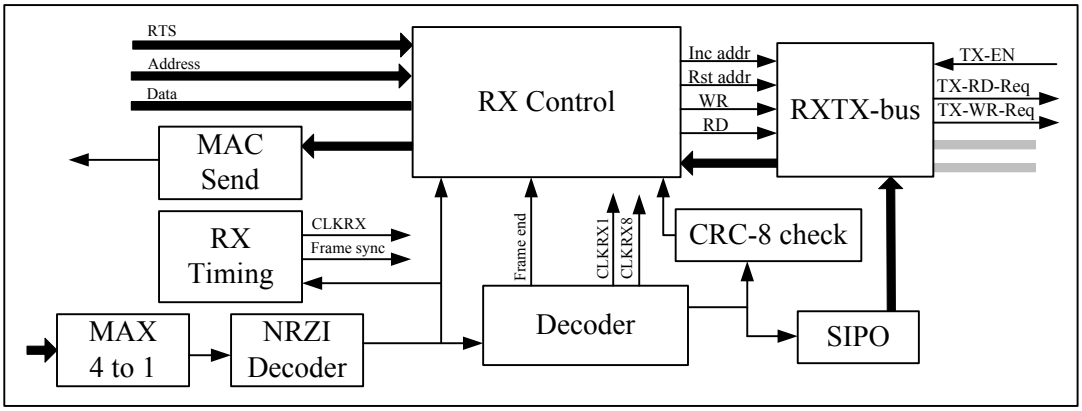


Figure 4.30: Block diagram of the receiver module.

Receiving starts if the *Signal Detector* module detects an RTS message and if *RX* is free. If clear, the *RX* replies with a CTS message generated by the *MAC Send* module and waits for *frame sync* bits to be detected by the *RX Timing* circuit. The *Decoder* is responsible for separating the synchronization bits. Regardless of the quality of the information conveyed, all received data is stored in the buffer through the *RXTX-bus*. In fact, it is at the end of reception that the *CRC-8 check* determines the validity of the received data. If this module confirms reception without error, then the *RX* changes the status of the corresponding buffer segment notifying a successful reception and then releases the buffer. Otherwise, the *RX* module just releases the buffer without further action.

4.4 Experimental Results

This section presents and discusses the experimental results obtained with the actual implementations of the SNs, BS and CPM pictured in Fig. 4.31. The supply voltage, clock frequency, and FPGA usage are summarized in Table 4.2. For the purpose of performance analysis, the communication nodes were first connected together with normal twisted wires and then the same experiment was repeated but now using conductive yarns for the interconnect. Signals flowing in the communication lines were analyzed and measured with an oscilloscope.

4.4.1 Communication in the MAC layer

Figure 4.32(a) shows the signals on the line, including both the *TX* (sender node) and *RX* (receiver node) signals, when two sensors are connected with normal twisted wire and communicating at a 2 Mbps data rate. Fig. 4.33 shows more details at 9 Mbps. The signal level at the output is between 0 V and 1.5 V. *RX* and *TX* signals can be seen separately in Fig. 4.32 (b and c). This figure also shows the complete MAC layer message including RTS, CTS, ACK and data packet frames with synchronization and start bits (waveform c). The transmitted packet includes 15 bytes all with 0x00 value (for this example) to highlight the synchronization bits in a packet including a long string of 0s. The trailer contains the CRC-8 checksum that is generated in the MAC layer.

Table 4.2: Characteristics of the communication circuit and its FPGA implementation

Parameter	Value
Model	Actel IGLOO AGLN125
Logic cell utilization	2509 of 3072 (81.7 %)
Internal memory	2 kB
I/O supply voltage	3.3 V
Core supply voltage	1.5 V
Clock frequency	16 MHz
Data rate	4 Mbps

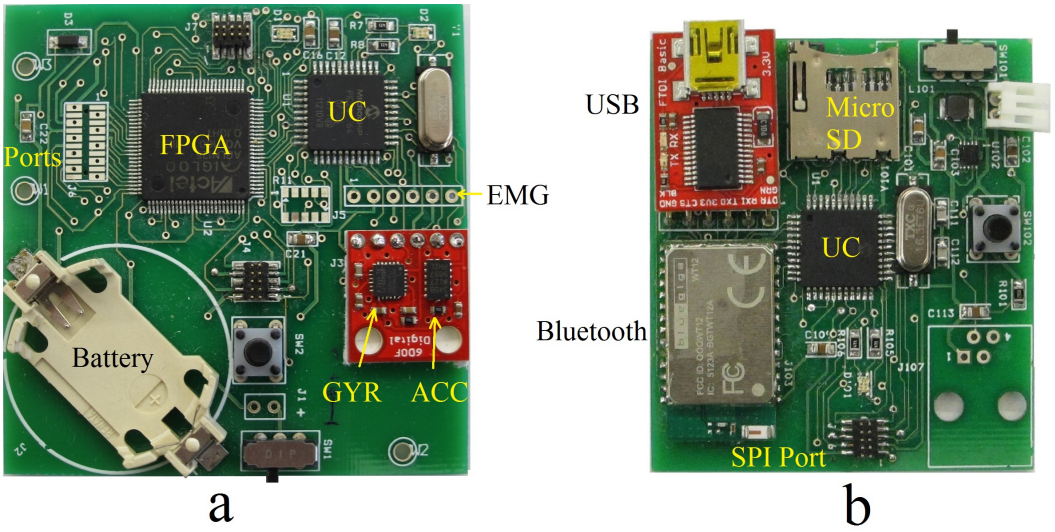


Figure 4.31: Picture of the prototype; a) SN and BS ($6 \times 6\,\text{cm}^2$), b) CPM ($5 \times 5.6\,\text{cm}^2$).

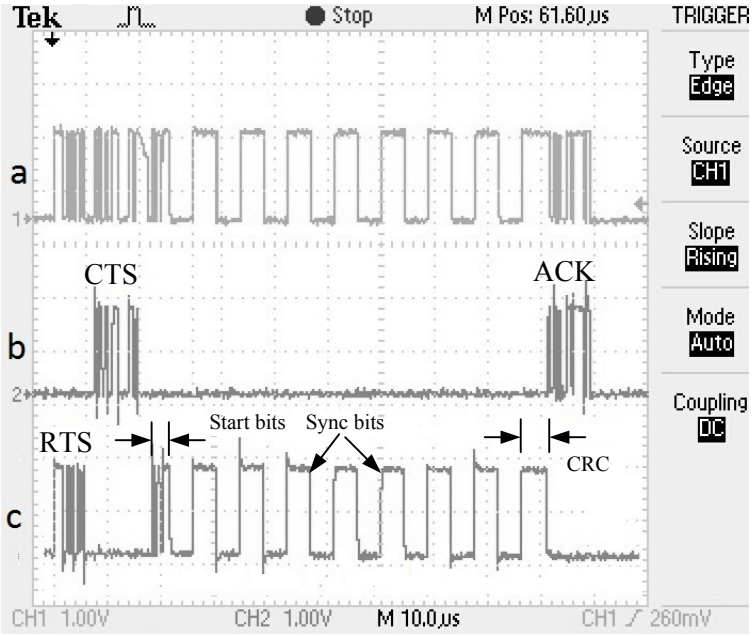


Figure 4.32: Encapsulated data packet in MAC layer: a) line, b) signals generated by receiver, c) signals generated by transmitter.

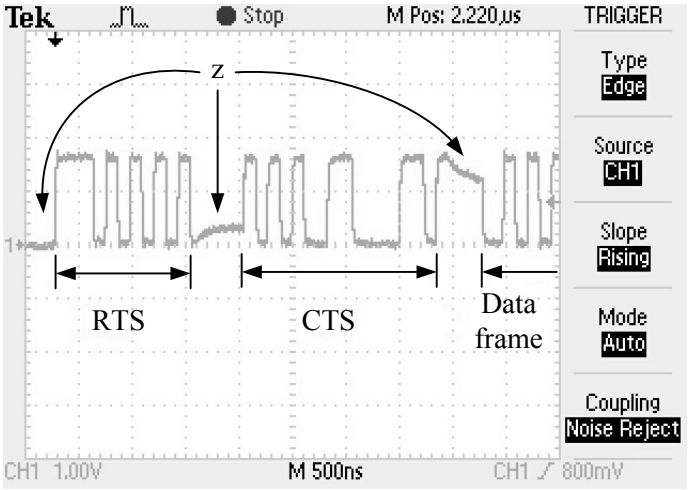


Figure 4.33: Signals over the line.

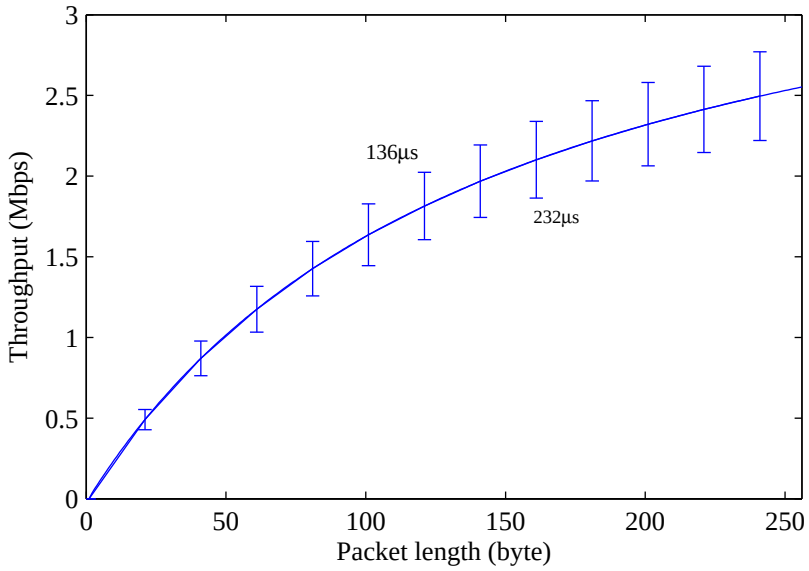


Figure 4.34: Node-to-node throughput as a function of packet length for a 4 Mbps data rate and T_{idle} in the range ([136 µs, 232 µs])

It should be noted that the upper layers (network layer and above) do not implement any error checking mechanism. In fact, extra error checking mechanisms are not necessary in the upper layers, because if any error is detected at the MAC level, the packet will be dropped before being handed over to the network layer and upper. Furthermore, including other checking structures would increase the packet overhead.

Assuming Poisson channels, the node-to-node or MAC level throughput is given by Eq. (4.34). Figure 4.34 depicts the throughput calculated with parameter values obtained from circuit operation and determined using Eq. (4.34) for $\lambda = 4$ Mbps. Obviously, throughput increases with packet length and reaches 2.55 Mbps@256 bytes, which is 64 % of the data rate. Throughput is mainly influenced by T_{idle}, which was measured to be 184 µs@16 MHz on average ([136 µs, 232 µs]). In the Chapter 6 it will be shown that T_{idle} due to packet serving time can be significantly reduced by moving the routing of packets from the microcontroller to the FPGA or IC, resulting in a better throughput as well.

4.4.2 Routing

To evaluate the routing performance of the circuits with the SRMCF protocol, a network containing three SNs and a BS, as shown in Fig. 4.35, was used. The bold lines show the minimum cost path values between SN3 and BS defined during the network setup phase (as described in Chapter 3).

The first experiment was designed to evaluate the effectiveness of the network in routing a packet sent from the BS to a SN. There are two intermediate nodes between the BS and the destination node SN3. Figure 4.37 illustrates the packets at different points of the network.

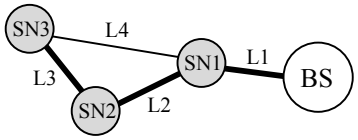


Figure 4.35: A network contains three SNs and BS.

As mentioned before and shown in Fig. 4.36, the length of the packet sent from the BS to a SN is variable and decreases as it passes through intermediate nodes. In Fig. 4.36, waveform L1 shows the packet after the BS, waveform L2 the packet after the first intermediate node, and waveform L3 after passing the second node.

Another experiment was performed to evaluate the routing delay for transmissions from a SN to the BS node. Again, there are two intermediate nodes between the SN and BS. Figure 4.37 shows the packet Pa (generated by SN3 with a payload of 128 bytes) as it is routed through the network. As can be observed, the delay to the first intermediate node is almost 150 µs (packet serving time), which is the same for the second. The end-to-end delay from SN3 to BS is 2.2 ms. The routing process time was measured to be in the range from 136 µs to 232 µs (with an average of 184 µs).

Figure 4.37 also shows that the signals are in a different polarity in the absence of communication between the SNs. Since NRZI is being used as the line coding scheme, the polarity of the signal is of no importance. The transmitted data indicated with an *M* in this figure shows two coexisting transmissions involving a common node. A time interval can be identified where SN3 is sending information to SN2, while a communication link is also active between SN2 and SN1. This shows the ability of the developed SNs to simultaneously send and receive data as described earlier.

4.4.3 Power consumption

The power consumption of the circuit as a function of data rate is depicted in Fig. 4.38. The measurement of power consumption in idle mode indicates that the power expenditure for data rates set to be below 512 kbps is 1.5 mW. It increases to 1.74 mW at 1 Mbps, 1.95 mW at 2 Mbps, and 2.7 mW at 4 Mbps. The increased power consumption in idle mode is due to the signal detector module, whose clock signal must be increased according to the data rate even in idle mode.

In order to operate at data rates above 4 Mbps the internal PLL of the FPGA needs to be activated, which increases the power consumption. To attain the maximum data rate (9 Mbps), the PLL output frequency must be set to 70.993 MHz, increasing the power consumption to 7.92 mW.

The power consumption during communication increases to 3.5 mW at 4 Mbps and 9 mW at 9 Mbps in TX mode; in RX mode the power consumption is 4 mW at 4 Mbps and 10 mW at 9 Mbps. Usually wired systems consume less power than wireless counterparts. The values reported in [PB10a] for Zigbee (30 mW, 250 kbps) and Bluetooth (100 mW, 3 Mbps) indicate that the proposed circuit consumes substantially less power than wireless alternatives.

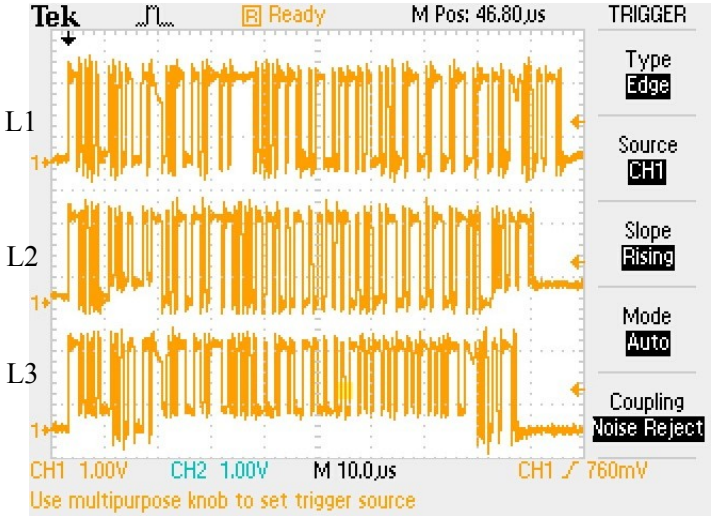


Figure 4.36: Routed packet at different points of the network from the BS to the destination SN.

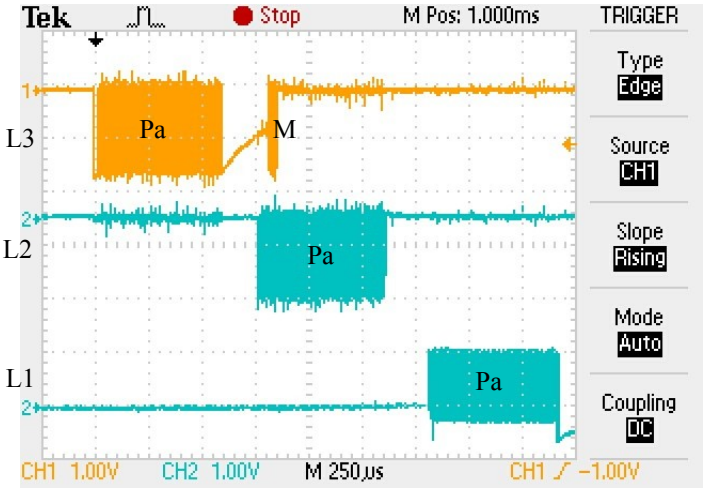


Figure 4.37: Data transmission from SN3 to BS observed at different points of the network (see Fig. 4.35 for line identifiers)

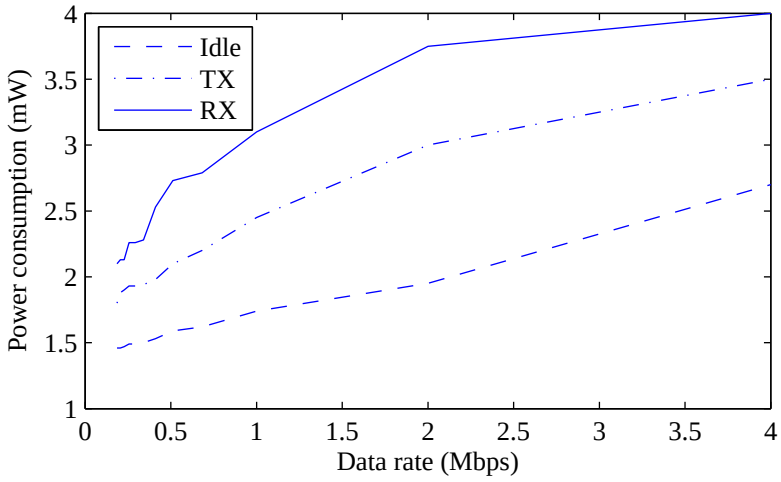


Figure 4.38: Power consumption in terms of data rate (VCC = 1.5 V).

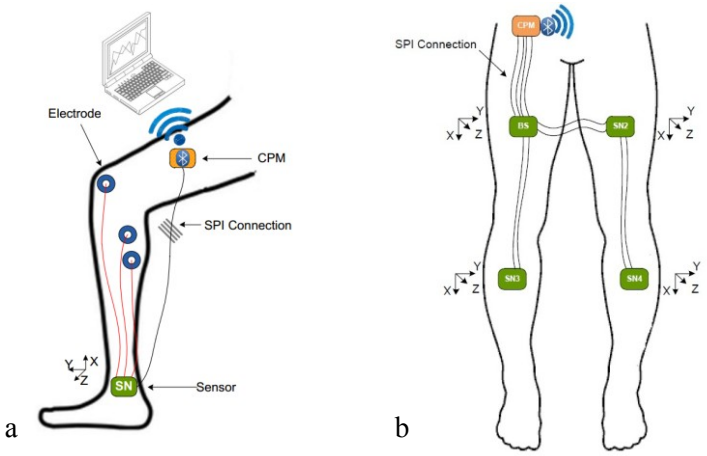


Figure 4.39: a) Setup for sEMG, b) Sensor node placement

4.4.4 Data acquisition examples

Figure 4.40 shows the acceleration, angular movement and sEMG signals captured with the SN shown in Fig. 4.39(a) , while the test subject makes four steps. The software module for displaying the data was developed by Ruben Dias for the ProLimb project. All data transfers are handled by the BAN described in this dissertation. The sampling frequency of the accelerometer and gyroscope is 50 Hz with 16-bit resolution in the three dimensions; the sEMG signal is sampled ay 1 kHz with 16-bit resolution. In these experiments, Ag/AgCl electrodes were used. Muscle activity is clearly identifiable in the sEMG signal. Notice the delay between sEMG signal and kinematic activity.

In the second experiment, data is acquired by four sensors connected to the lower limbs as shown in Fig. 4.39(b). Figure 4.41 shows the data acquired from the gyroscopes located as shown in Fig. 4.39(b) while the subject makes four steps with the right leg and three with the left.

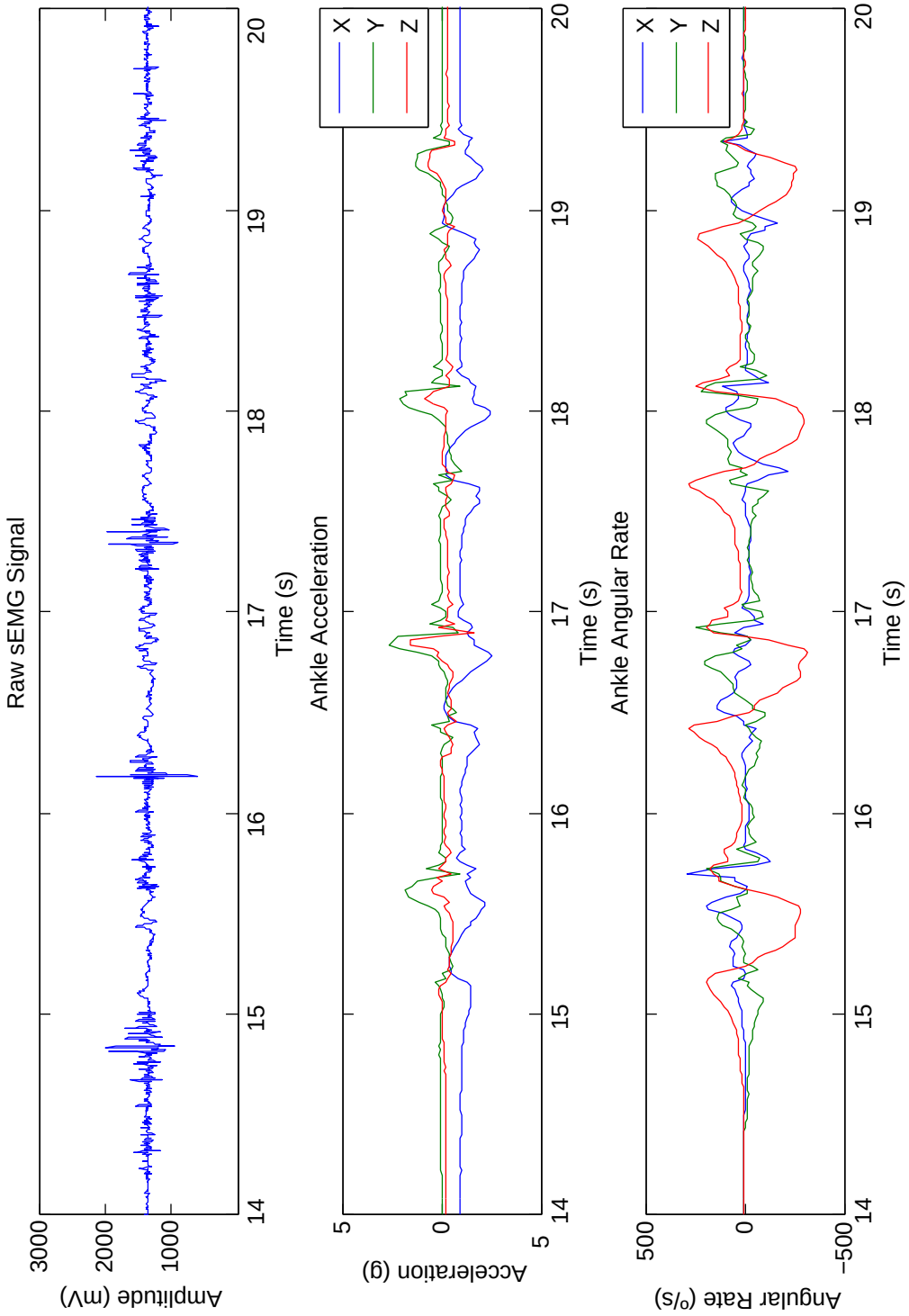


Figure 4.40: Acquired electromyographic and inertial signals

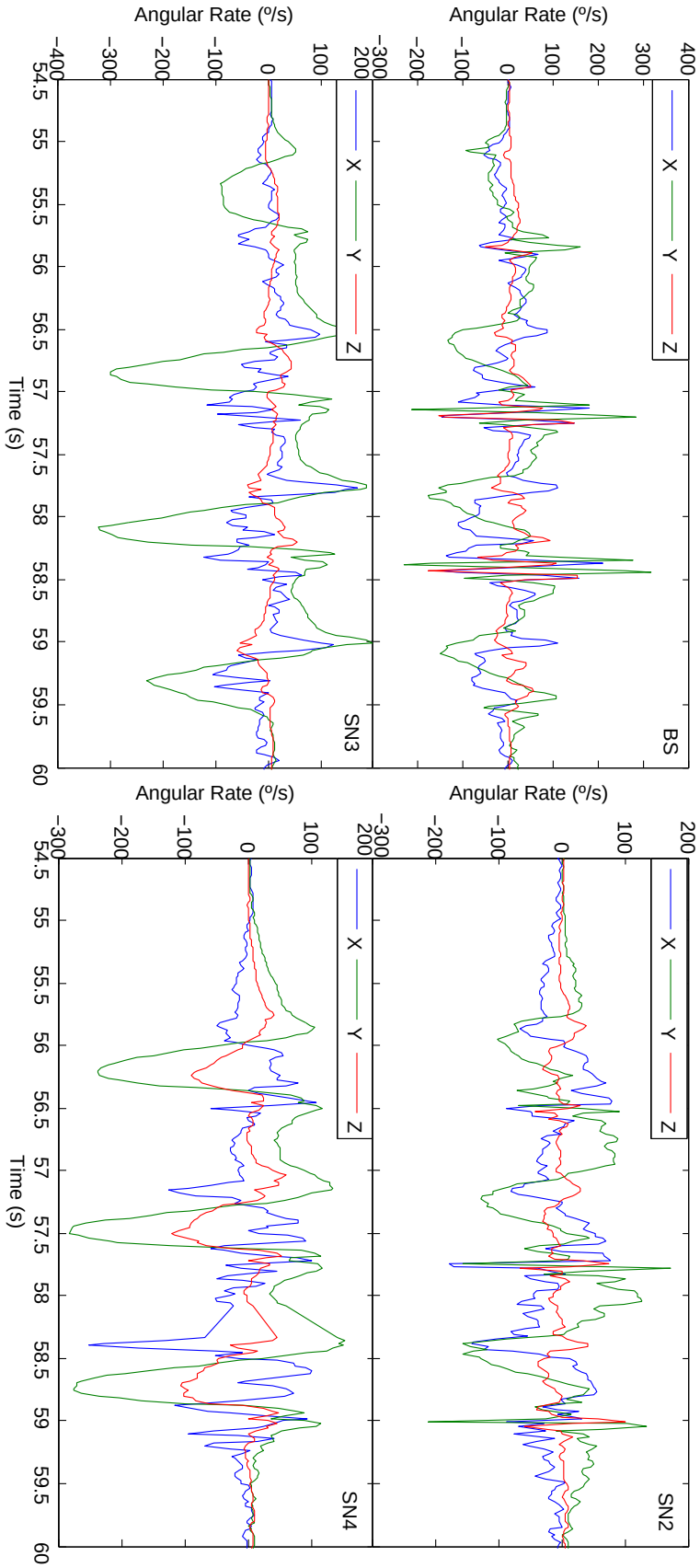


Figure 4.41: Example of data acquired in real time from the gyroscopes

4.4.5 Network of Sensors on Textile

Figure 4.42 depicts a network on textile produced by André Whiteman Catarino (from Universidade do Minho). In Fig. 4.42, the bold lines represent the selected paths (the others are redundant). All the SNs are connected to sEMG electrodes and are also equipped with accelerometer and gyroscope. For the network of Fig. 4.35, CRC errors have been found to be extremely rare, because of the very short range communication over wires. So, for this experiment, the nodes have been set to not send ACK message to sender, but the other configuration parameters and sampling rate are the same as in previous measurements.

Figure 4.43 shows the average number of the packets saved in buffers as a function of packet length (Eq. (4.38)) in BS, SN2 and SN5 while generating 20.8 kbps of data in each node (16 kbps by the sEMG, 2.4 kbps by the accelerometer and 2.4 kbps by the gyroscope). The service rate (μ) is

$$\mu = \frac{drate}{P_l} + T_{idle}, \tag{4.42}$$

where *drate* is data rate (4 Mbps), P_l is packet length and T_{idle} is the average time to process the received packet (184 µs for SNs and 0 s for BS). The results shows that the average number of the packets in the buffers is always small and the packet loss probability due to buffer overflow shown in Fig. 4.44 (and calculated according to Eq. (4.41)) is very small. So, the network is able to collect data from the SNs with high reliability.

The system performance under high traffic with the same configuration was also evaluated. For that, each SN generates 640 kbps, which is almost 31 times more than the previous value. Figures 4.45 and 4.46 depict the average number of packets in the buffer and packet loss probability, respectively. The performance of the network decreases especially for small packet lengths, as expected. So, to improve network performance with high data traffic, increasing the length of the packet can be a solution. This could be achieved by more sensor data in a single packet.

The average end-to-end delay for both low and high traffic (20.8, 80, 160,...,640 kbps) in terms of packet length is portrayed in Fig. 4.47. In low traffic conditions, the delay increases linearly with packet length. In high traffic the behavior is different because of the effect of T_{idle}. It always affects the network performance, but in high traffic with low packet length its effect is more significant and causes a notable increase in the end-to-end delay. Nevertheless, in all conditions, the average delay does not exceeds 4.3 ms.

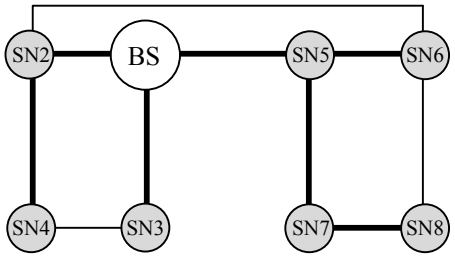


Figure 4.42: A network of SNs embedded in textile used for data acquisition from the lower limb.

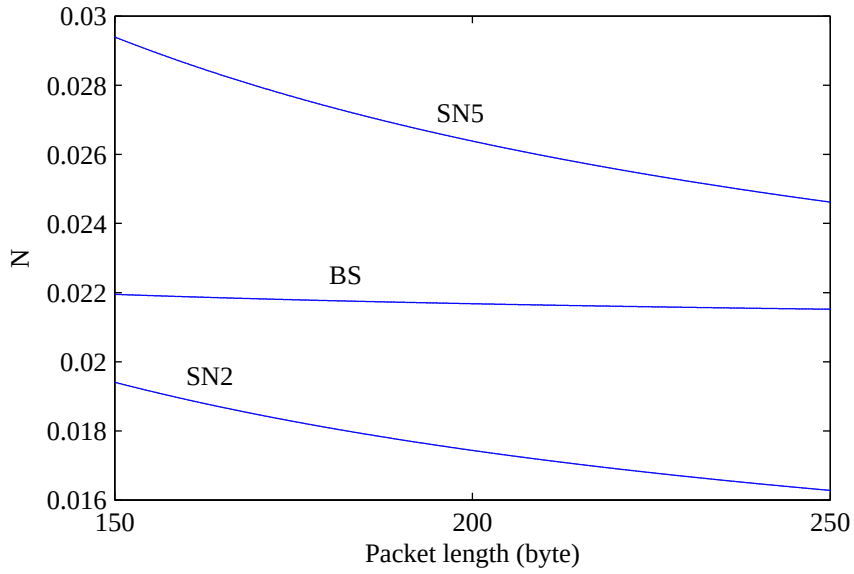


Figure 4.43: Average number of the packets in buffers with 20.8 kbps traffic per node

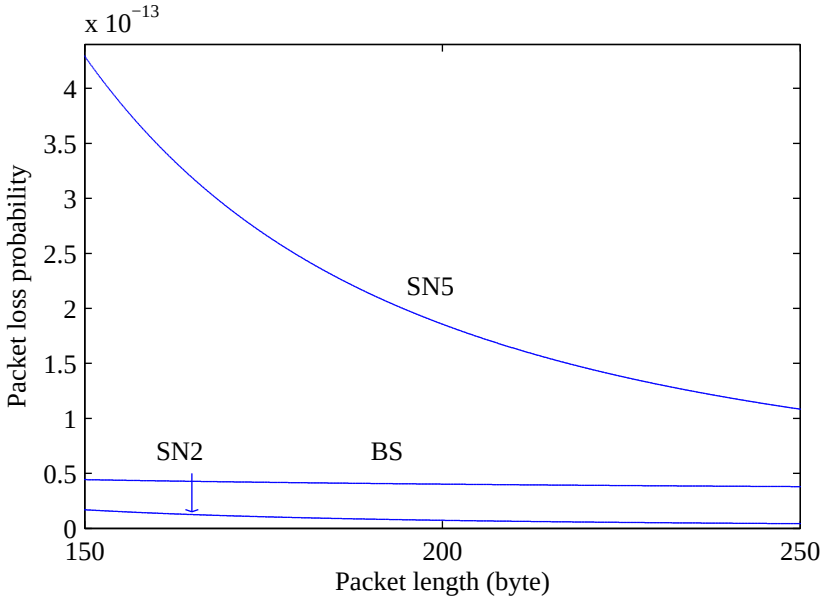


Figure 4.44: Packet loss probability with 20.8 kbps traffic per node

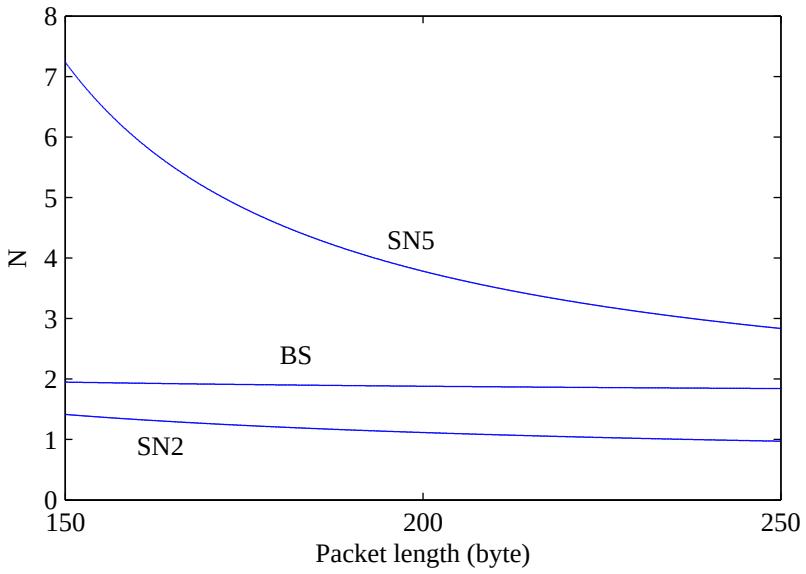


Figure 4.45: Average number of the packets in buffers with 640 kbps traffic per node

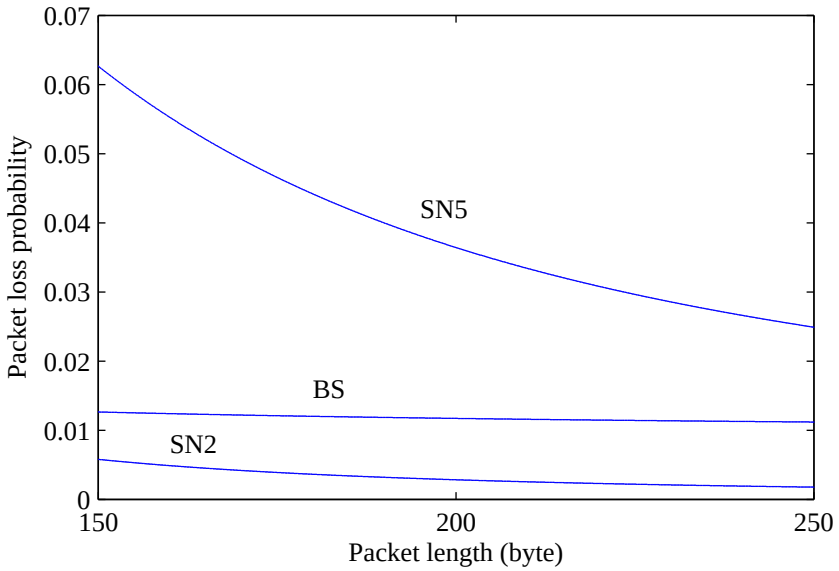


Figure 4.46: Packet loss probability with 640 kbps traffic per node

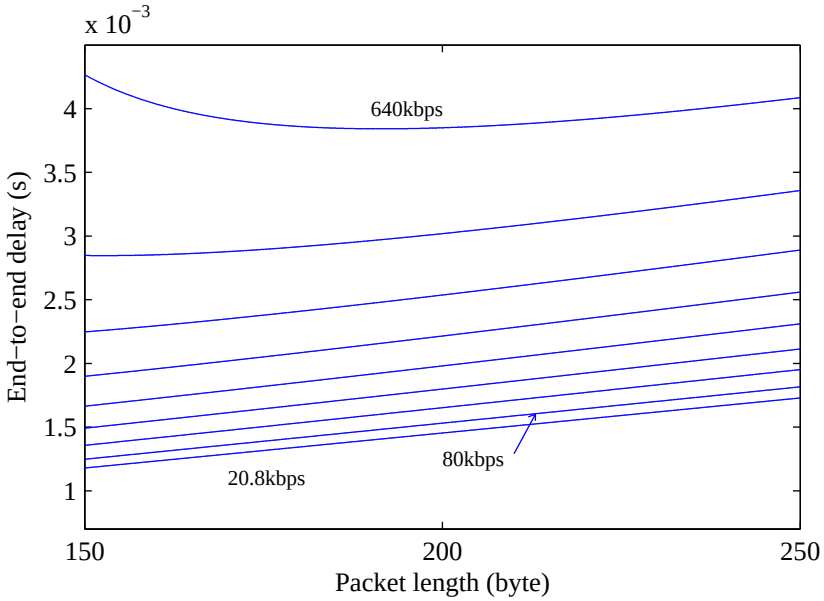


Figure 4.47: Average end-to-end delay.

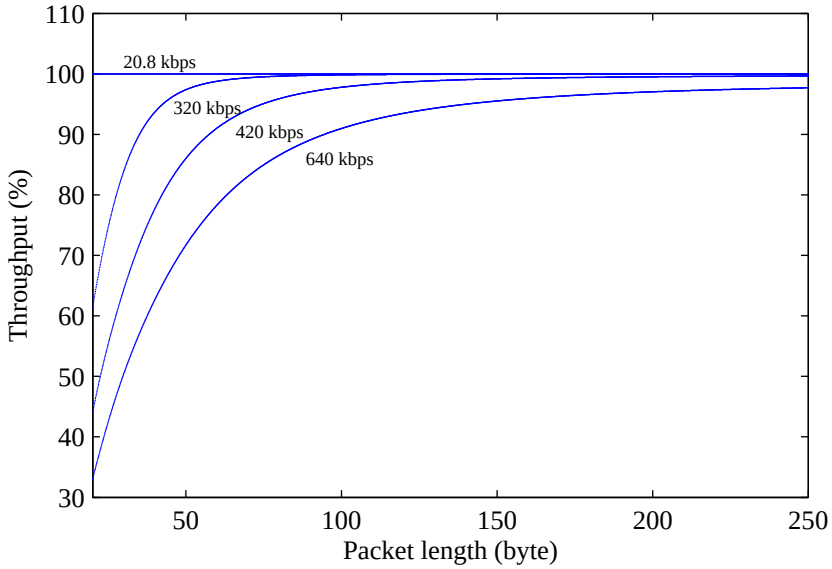


Figure 4.48: Throughput of the network in terms of the packet length.

The SN-to-BS throughput (as the percentage of the total generated traffic) of the network in terms of packet length for different traffic conditions is shown in Fig. 4.48. Obviously, throughput increases with packet size. Nevertheless, the throughput improvement does not have linear relation with packet length, e.g. for 640 kbps the throughput increases from 35 % to 90 % for packet lengths of 25 B and 100 B, respectively, which represents an increase of 2.57× in throughput. However, throughput improves only 1.07× (from 90 % to 96 %) when the packet length changes from 100 B to 250 B. In any case, it can be concluded that throughput improves as the packet length increases or traffic decreases and it would be better to generate the packets with long size.

4.5 Conclusion

The first prototype of a wearable system and the results obtained from actual realized SNs have been presented. The system performed as expected and was validated experimentally. The described circuit and sensor network confirm that mesh topologies can be used in systems with high number of sensors and high data-rate. Although the sensors consume a small amount of power, the number of intermediate nodes in the path between BS and SNs should be kept as small as possible. Buffering packets in intermediate nodes aggravates the end-to-end delay and also augments power consumption caused by the extra packet processing needed for routing. Performing the routing at software level in the microcontroller also increases the end-to-end delay, mainly due to the time required for reading packets via the SPI port.

The main contributions of this chapter are:

1. Design of a wearable system, including the CPM and wearable network in a mesh topology.
2. Analysis of the network behavior assuming a Poisson process with exponential arrival intervals: throughput, end-to-end delay, probability of packet loss.
3. Network stack implementation including physical, MAC, network, middlware and application layers.
4. Hardware design of SN, BS and CPM circuits.
5. Implementation of the physical and MAC layers on FPGA.
6. Implementation of the SRMCF routing protocol, middleware and application layers on a microcontroller.
7. Experimental evaluation.

In Chapter 6 a different end-to-end communication method to improve the performance is presented. The idea is to use circuit or hybrid circuit and packet switching instead of pure packet switching; additionally, routing will be performed by an IC designed to that end, which eliminates the need to have a microcontroller for implementing that functionality.

Chapter 5

Synchronization Protocols

In all kinds of networks and distributed systems, such as sensor networks, providing synchronization in the system is important and critical. In general, synchronization refers to the process of making events occur at the same time in different parts of a distributed system. Successfully recovering data in the receiver side of digital communication systems, totally depends on synchronization between the receiver and transmitter. On the other hand, in sensor networks, like in any other data acquisition system, it is often necessary to record timing information with the data. For these systems, the challenge is to maintain performance and precise synchronization between the nodes using only limited resources.

This chapter contains two sections. In the first a CDR with some specific features used in circuits is analyzed and the implementation of the actual circuit is explained. In the second section, a high precise time synchronization protocol implemented at hardware-level is presented. In both sections the reported results have been obtained from simulation and also from the real circuits.

5.1 Clock Synchronization

This section describes a CDR circuit based on open-loop synchronization for use in wearable systems including the circuit presented in Chapter 6. Sensor nodes are able to handle simultaneous communication with other nodes. For that purpose, each port is equipped with an independent CDR circuit. Since there are several instances of the CDR in each node, the aim of this work is reducing the circuit size as much as possible. Its main contributions are:

1. The design of a very small, fully digital, fast-lock clock CDR circuit for use in resource-constrained sensor nodes;

2. The implementation of the design in two different technologies: Actel FPGAs and a 0.35 µm CMOS IC;

3. The experimental evaluation of the proposed design using fabricated IC prototypes.

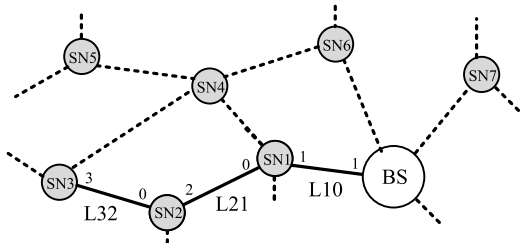


Figure 5.1: A mesh-based body area network. The base station (BS) gathers the data collected by the sensor nodes (SN). All nodes are connected by conductive yarns embedded in textiles.

ta

5.1.1 Motivation

As presented in previous chapters, the wearable system includes a number of sensor nodes connected to each other by conductive yarns in a mesh topology, as shown in Fig. 5.1.

From the standpoint of the network, each SN is a four-port router with bidirectional links to other SNs. The SNs are equipped with a local crystal oscillator. When sensors communicate, the receiver node must be able to synchronize its clock to the incoming data. Because there is no global synchronization, a local synchronization method is necessary. The following factors were taken into account in the design of the clock synchronization circuit:

- **Size:** In the architecture described here, SNs are multitasking network devices that can handle several transmitting or receiving connections simultaneously. In the prototype implementation, each SN has four independent CDR circuits. In these situations, it is important to have small circuits, since it is impractical to have several PLLs/DLLs and FIFO memories in resource-constrained sensor nodes: for instance, smaller FPGAs, like the IGLOO nano family [Mic13], have at most one PLL.

- **Energy consumption:** Sensor nodes are typically energy-limited systems. Traditional open-loop approaches need higher bandwidth to carry synchronization signals, increasing the energy consumption at the transmitting part. Nevertheless they usually consume less energy than the closed-loop ones, because of their simpler structure and absence of modules like VCOs or frequency synthesizers.

- **Bandwidth:** Although conductive yarns are similar to regular copper wires, they exhibit a much higher electrical resistance, leading to increasingly significant signal attenuation with increasing frequency. This drawback leads to the use communication mechanisms or modulations that reduce the necessary bandwidth. However, open-loop approaches based on band-pass filtering require clock spectral information to be present in the signal, thereby increasing bandwidth usage, since clock frequency is usually 2 times the data rate of baseband communication.

- **Internal node synchronization:** Even if precise clock recovery is achieved, synchronization between the core of the system and the recovered clock signal should be taken into

account. Each sensor is a multitasking device that must be able to communicate with all its neighbors simultaneously. The core of the system must be able to process data from different sensors at the same time (using different recovered clocks) and therefore some kind of internal synchronization is required. FIFO memories or elastic buffers can be used to do internal synchronization, but they increase circuit size.

The open-loop synchronization method introduced in this work is a trade-off between the aforementioned factors. Like open-loop approaches, it has a less stable sampling point than provided by closed-loop methods, but it is able to recover data with high accuracy and very low BER. Because of the small number of elements needed to implement the circuit, size and energy consumption are much smaller than for any oversampling or close-loop circuits. Close-loop circuits need some preamble signals to lock on the incoming signal and to generate a stable clock, but the circuit presented here only needs one transition in the incoming signal to start clock synchronization. In contrast to some open-loop circuits, the proposed circuit does not use a filter and there is no need for the incoming signal to include clock spectral information. Therefore, the new circuit architecture utilizes less bandwidth than open-loop systems based on band-pass filtering. Finally, the proposed circuit uses the system clock as reference, so the receiver circuitry is always synchronized with the core of the sensor node without the help of elastic buffers or FIFO memories.

5.1.2 Synchronization Method

This section describes the proposed clock synchronization method for CDR without using feedback loops or embedded spectral information. We also determine the conditions for correct operation: the data rate must not exceed half the clock frequency.

5.1.2.1 Autonomous Clock Oscillators and Synchronization

Crystal quartz resonators have very low variability, slight temperature dependency and present very high Q-factors (quality factor), they are the preferred devices for generation of stable waveforms [SAHW90]. Therefore, we assume that SNs have an autonomous clock generator that provides a signal with the same nominal frequency and drives the whole system, including transmitter and receiver modules.

The output of an ideal oscillator is given by Eq. (5.1):

$$C(t) = A \sum_{k \in \mathbb{Z}} p(t - kT) \tag{5.1}$$

Here, A denotes the signal amplitude, T is the clock period and $p(t)$ is the unit square pulse with $T/2$ width. In addition to inter-device variations, a real clock waveform also suffers from noise, temperature and power supply variations, and other ambient effects. Equation (5.2) shows the real local clock waveform at sensor node n,

$$C_n(t) = A \sum_{k \in \mathbb{Z}} p(t - kT_n - \tau_n(t)), \tag{5.2}$$

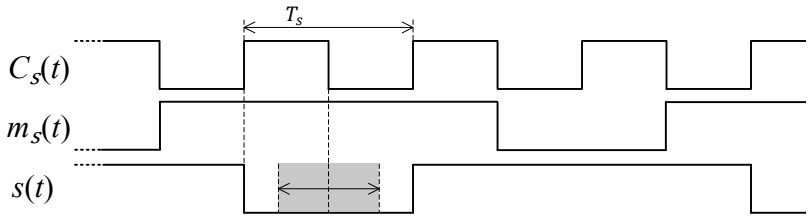


Figure 5.2: NRZI signal $S(t)$ generated from $m_s(t)$.

which includes phase noise $\tau(t)$ (with zero mean value) and clock period T_n. The term $\tau(t)$ causes the signal to shift randomly over time and generates jitter. The frequency of each oscillator ($f_n = 1/T_n$) is a random variable that is uniformly distributed around the nominal frequency with accuracy of a few ppm.

Consider the case where an SN is communicating with its neighbor in base band (e.g., SN1 and SN2 in Fig. 5.1). In general, recovery of clock timing depends on the line encoding. For example, in Manchester coding there is at least one transaction for each transmitted bit [Mad08]. In other words, spectral information of clock signal is transferred to the receiver, which can easily recover the timing information. The drawback of such an encoding method is the need for higher bandwidth. As discussed in Section 5.1.1, the present work does not use this approach.

In contrast, encoding methods like NRZ or NRZI need half of the bandwidth for the same amount of data, but the receiver has to use a more complex method to extract timing information from the incoming signal [Mad08]. Figure 5.2 shows an NRZI-encoded signal $S(t)$ produced by encoding the signal $m_s(t)$ using clock $C_s(t)$ with period T_s (subscript s denotes to the sender node). The dashed line indicates the optimum sampling time at the receiver, which is in the middle of the data interval. For sampling to occur at that time the phase difference between receiver and transmitter clocks must be zero, but noise and jitter may cause the receiver not to take a sample at the optimum instant. However, as long as sampling occurs around the optimum point (the gray region in Fig. 5.2), it is possible to recover data successfully. The boundaries of the sampling range depend on the signal jitter.

The autonomous system oscillator at the receiver has a fixed frequency, which is m times the data rate, and generates a recovered synchronized clock signal from the system clock by tracking the transitions of the incoming signal. The recovered clock is a set of pulses with frequency equal to the data rate (m times slower that the system clock); the CDR's task is to shift the pulses so that the active edges are always in the sampling range for correct data recovery. The general idea is to detect if an edge of the incoming signal occurs close to an active edge of the local clock signal. If the incoming edge occurs just before an active clock edge, the next clock edge is delayed by half the data period; if the incoming edge occurs just after the active edge, the following one is advanced by the same amount of time.

Considering the clock signal given by Eq. (5.2), the transmitter's clock phase $\varphi_s(t)$ is given by

$$\varphi_s(t) = 2\pi \frac{t_s(t)}{T_s}, \tag{5.3}$$

where $t_s(t)$ is

$$t_s(t) = t - kT_s - \tau_s(t). \tag{5.4}$$

The receiver's clock phase $\varphi_r(t)$ is given by a similar equation. Using the index s for quantities associated with the transmitter and index r for those associated with the receiver, the phase difference between sender and receiver is given by

$$\Delta\varphi(t) = \varphi_s(t) - \varphi_r(t) = 2\pi \left(\frac{t_s(t)}{T_s} - \frac{t_r(t)}{T_r} \right). \tag{5.5}$$

Assuming that the mean values of the phase noise $\tau_s(t)$ and $\tau_r(t)$ are zero and that their values in any given cycle k are negligible, then Eq. (5.5) can be rewritten as

$$\Delta\varphi(t) \approx 2\pi t \left(\frac{1}{T_s} - \frac{1}{T_r} \right) = 2\pi t(f_s - f_r) = 2\pi t \Delta f. \tag{5.6}$$

Equation (5.6) shows that the $\Delta\varphi(t)$ changes linearly over time with a constant slope $2\pi\Delta f$.

As a preliminary step, we evaluate the effects of $\Delta\varphi(t)$ on synchronization when the receiver clock generator frequency is equal to data rate. Assume that receiver takes samples of data at the negative edge of the clock, and that at $t = 0$ the receiver clock is in synchronization for sampling at the optimum point ($\Delta\varphi = 0$). Over time, $\Delta\varphi(t)$ changes until it is out of the sampling range at t_1. Therefore,

$$\Delta\varphi(t_1) = \Delta\varphi_{sp} \Rightarrow t_1 = \frac{\Delta\varphi_{sp}}{2\pi\Delta f}, \tag{5.7}$$

where $\Delta\varphi_{sp}$ is the maximum phase difference between $C_r(t)$ and $C_s(t)$ so that data is correctly detected (negative edge of receiver clock is in the gray area of Fig. 5.2). For $t > t_1$, the clock receiver is out of synchronization and needs to shift the negative edge to be in the sampling interval again. As mentioned in Section 5.1.1, synchronization between peripheral modules and system core must also be maintained without using FIFO memories. Any partial shifting of the clock (as done in DLLs) will make it difficult to keep the synchronization with the system core and will require FIFO memories and a more complex circuit. To keep the circuit simple, shifting of the clock signal must be done by a full clock period to the right or to the left. However, shifting the clock signal by one cycle keeps the relative sampling time unchanged. Therefore, under these constraints, the receiver cannot detect data correctly if its frequency is the same as the data rate.

5.1.2.2 Synchronization Using a Faster System Clock

One way to overcome that problem is to increase the system clock frequency and generate the receiver clock by division. This situation is shown in Fig. 5.3.

Signal $C_2(t)$ is the system clock and has twice the frequency of $C_r(t)$, which matches the data rate. The $C_r(t)$ signal is generated locally in the receiver by division of $C_2(t)$ at the raising edge and synchronized with the incoming signal. The arrows on the falling edges of $C_r(t)$ indicate the sampling times and the gray interval marks one specific sampling range.

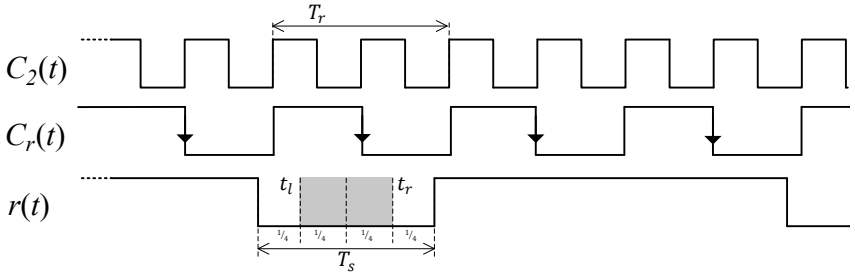


Figure 5.3: Receiver clocks and incoming signal.

As in the previous case, whenever the falling edge of $C_r(t)$ would fall outside the sampling range, the CDR circuit has to detect the situation and shift the edge to the correct sampling range (cf. Section 5.1.3). In this case, the shifting of $C_r(t)$ will be by one $C_2(t)$ clock period or $\frac{1}{2}T_r$. For example, if the frequency of $C_r(t)$ is a little lower than the data rate, then $C_r(t)$ will be moved to the right in relation to the incoming data. When the skew of the optimum sampling point reaches $\frac{1}{4}T_r$, the receiver shifts $C_r(t)$ from point t_r in Fig. 5.3 to point t_l on the left. Therefore, the phase difference between transmitter and receiver clock will be kept in the range

$$\frac{-\pi}{2} < \Delta\varphi(t) < \frac{\pi}{2}. \tag{5.8}$$

5.1.2.3 Impact of Jitter

After shifting, $C_r(t)$ still has to be in the sampling range. Therefore, the sampling range must be larger than the $\frac{1}{2}T_r$, resulting in the condition

$$R > \frac{T_r}{2}, \tag{5.9}$$

where R is the acceptable sampling range to detect data correctly, which should be at least 50 % of the data period. This range may not be acceptable for some other communication systems, specially for those with small SNR or high BER. In systems with low BER, like the wearable system mentioned before, the range is acceptable. The mechanism for detecting when to shift $C_r(t)$ so as to avoid loss of synchronization is described in Section 5.1.3.

In the absence of jitter, sampling can be done at any instant inside the data period and R is upper bounded by T_r. However, the random fluctuations of the incoming signal due to jitter may affect the generated clock $C_r(t)$ and cause the sampling point to be limited as shown in Fig. 5.4. Combining the upper bound condition and the impact of jitter, the sampling range of Eq. (5.9) must satisfy

$$T_r - 2|J| > R > \frac{T_r}{2}, \tag{5.10}$$

where J denotes the total jitter of the incoming signal (including both random and deterministic

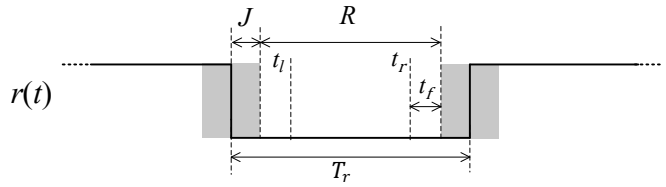


Figure 5.4: Boundaries of sampling range in the presence of jitter.

jitter). In this case, Eq. (5.8) becomes

$$\frac{-\pi}{2} - \varphi_J < \Delta\varphi(t) < \frac{\pi}{2} + \varphi_J, \tag{5.11}$$

where φ_J is maximum phase noise caused by jitter. For data to be correctly detect data correctly, any jitter must occur outside the sampling range.

Since any shifting of the incoming signal by jitter should not alter the sampling range, jitter should be less than $\frac{1}{2}(T_r - R)$. However, jitter also shifts $C_r(t)$. To take account of the effect of jitter on $C_r(t)$, the overall effect of jitter on the sampling range must be multiplied by two. Therefore, correct operation implies that the jitter must satisfy

$$|J| < \frac{1}{2}\left(\frac{1}{2}(T_r - R)\right). \tag{5.12}$$

Combining Eqs. (5.9) and (5.12) gives

$$|J| < \frac{1}{8}T_r. \tag{5.13}$$

5.1.2.4 System Clock Frequency

In general, if the clock frequency of the oscillator at the receiver is m times higher than the data rate, then a shift by one clock cycle will correspond to $\frac{1}{m}T_r$. In this case, the phase difference between transmitter and receiver clock is given by

$$\frac{-\pi}{m} - \varphi_J < \Delta\varphi(t) < \frac{\pi}{m} + \varphi_J. \tag{5.14}$$

Again, the sampling range must obey the condition

$$T_r - 2|J| > R > \frac{T_r}{m}. \tag{5.15}$$

As explained for $m = 2$ and from Eqs. (5.12) and (5.15), the jitter must be bound according to

$$|J| < \left(\frac{m-1}{4m}\right)T_r. \tag{5.16}$$

Figure 5.5 shows the clock skew between sender and receiver for $m = 2$ and $m = 3$ for the situation when the sender's clock frequency is a little less than the receiver's ($\Delta f < 0$). The dotted

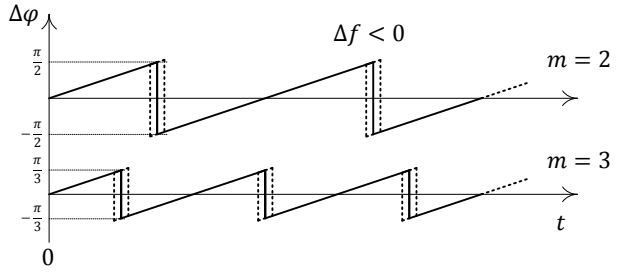


Figure 5.5: $\Delta\varphi(t)$ for $m = 2$ and $m = 3$ with $\Delta f < 0$.

lines show the jitter effect on $C_r(t)$. When $m = 2$, the clock skew slowly increases until $\Delta\varphi$ reaches $\pi/2$. Then the synchronizer shifts the receiver clock to the right, so as to have a phase difference of $-\pi/2$ to the incoming signal. For $m = 3$ the range is smaller.

5.1.2.5 Average of $\Delta\varphi(t)$

For each period of $\Delta\varphi(t)$, the average value of $\Delta\varphi$ can be calculate as the middle point of each line as

$$E\left[\tilde{\Delta\varphi}\right] = \frac{1}{2}\left(\frac{\pi}{m} - \frac{\pi}{m}\right) = 0. \tag{5.17}$$

For any arbitrary period starting from $\varphi 1$ we have

$$E\left[\tilde{\Delta\varphi}\right] = \sum_{i=\varphi 1}^{\varphi 1+2\frac{\pi}{m}} \Delta\varphi_i . p(\Delta\varphi_i), \tag{5.18}$$

where $p(\Delta\varphi_i)$ is the probability of $\Delta\varphi_i$. Considering that after each clock period the value of $\Delta\varphi$ changes by $2\pi\Delta t\Delta f$ (Eq. (5.6) with $\Delta t = T_s - T_r$) and the conditions from Eq. (5.8), we have

$$p(\Delta\varphi_i) = \frac{\Delta\varphi_i}{2\frac{\pi}{m}} = m\Delta t\Delta f \tag{5.19}$$

Regardless of the receiver and transmitter oscillator frequencies, $\Delta t\Delta f$ is constant, so that $f(\Delta\varphi_i)$ is the same for all points and

$$E\left[\tilde{\Delta\varphi}\right] = p(\Delta\varphi) \sum_{i=\varphi 1}^{\varphi 1+2\frac{\pi}{m}} \Delta\varphi_i = 0. \tag{5.20}$$

Equation (5.20) shows that the receiver always tries to keep the clock edge around the ideal sampling point. With jitter, the $E\left[\tilde{\Delta\varphi}\right]$ value will be the same, because jitter is due to random processes with zero mean value.

Increasing the receiver system clock frequency decreases the sampling range and increases the accuracy of sampling. However, this also increases energy consumption. For the wearable system where the proposed circuit is used, 50 % sampling range is acceptable, so the minimum value of $m = 2$ is used for the prototype circuit presented in the next section.

5.1.2.6 Number of input signal transitions

The CDR mechanism is based on tracking the transitions of the received signal. A long string of 1s or 0s (with a NRZ line code) or just a long string of 0s (with a NRZI line code) may cause loss of synchronization. To evaluate how long the CDR can be allowed to stay free running for $m = 2$, it is necessary to consider the worst-case situation after the last transition of the incoming signal: the recovered clock is exactly at the edge of the sampling range (t_r in Fig. 5.4) and will leave that range in the next period. The data is still correctly acquired if the sampling point is inside the safe zone shown in Fig. 5.4. This means that the sampling point may shift right by up to

$$t_f = \frac{T_r}{2} - \frac{R}{2} - |J| < \frac{1}{8}\tau. \tag{5.21}$$

Over N successive clock cycles, the accumulated shift $N|\tau_s - \tau_r|$ must stay below t_f:

$$N|\tau_r - \tau_s| < t_f < \frac{1}{8}\tau \Rightarrow N < \frac{\tau}{8|\tau_r - \tau_s|} \tag{5.22}$$

The frequency of oscillators is distributed around the nominal frequency and can be in the range of few parts per million (ppm) for a crystal quartz oscillator. To obtain the minimum value of N consider the worst case: $f_r = f(1 + p_c)$ and $f_s = f(1 - p_c)$, where p_c denotes to the frequency tolerance range. For the CDR circuit to work correctly, N must be

$$N < \frac{\frac{1}{f}}{8\left|\frac{1}{f_r} - \frac{1}{f_s}\right|} \Rightarrow N < \frac{1 - p_c^2}{16\,p_c} \tag{5.23}$$

A typical tolerance value for a quartz oscillator is ± 30 ppm, which leads to $N < 2083$. A well-stabilized RC oscillator may have a tolerance range of ± 1000 ppm, leading to $N < 6.25$. If the conditions on N are not satisfied by a specific implementation, a line code such as 8b/9b [Mad08] must be used to overcome the effect of a long strings of identical symbols. The prototype IC used to obtain the experimental results can be configured to use any of the line codes 8b/9b, 16b/17b, ..., 64b/65b.

5.1.3 The Synchronization Circuit

This section presents the implementation of a digital circuit that operates on the principles described in last section. The goal is to have a small size and fully digital circuit that can be completely implemented in an FPGA or digital ASIC. In order to keep the frequency of the local clock generator as low as possible, a solution corresponding to $m = 2$ is presented.

To generate a synchronized clock, the circuit has to meet the conditions of Eqs. (5.8) and (5.9). Figure 5.6 shows the block diagram of the circuit. At the receiver, a stable crystal quartz oscillator generates the system clock $C_2(t)$, whose frequency is twice the data rate and nominally equal to the clock frequency of the transmitter.

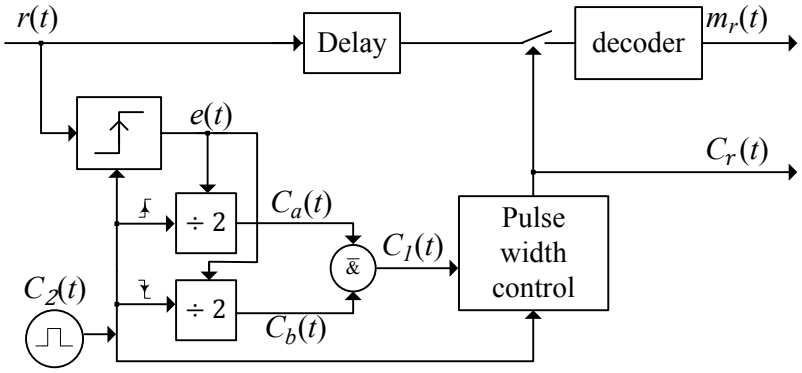


Figure 5.6: Block diagram of the circuit.

An edge detector determines both falling and raising edges of the incoming data signal $r(t)$ and generates the $e(t)$ signal shown in Fig. 5.7. Signal $e(t)$ goes high on any transition of $r(t)$ and goes low at the negative edge of $C_2(t)$. The width of the $e(t)$ pulses can vary from 0 to $\frac{1}{2}T_r$, depending on the phase difference between $r(t)$ and $C_2(t)$.

Two clock signals $C_a(t)$ and $C_b(t)$ are generated from $C_2(t)$ by division at the rising and falling edges, respectively. These have the same clock period T_r (equal to the unit interval of the system), but a phase difference of 90°. They are generated by D-type flip-flops triggered respectively by the rising and falling edges of $C_2(t)$. The generation of both signals is also controlled by $e(t)$ in order to be synchronized with the incoming $r(t)$: the $e(t)$ pulses are used to asynchronously reset the flip-flop that generates $C_a(t)$ and preset the flip-flop that generates $C_b(t)$. Because the negative edges of $e(t)$ are synchronized with the negative edges of $C_2(t)$, both $C_a(t)$ and $C_b(t)$ will always be synchronized with $e(t)$ as well. This is indicated by the arrows in Fig. 5.8.

Signals $C_a(t)$ and $C_b(t)$ are combined by a NAND gate to produce $C_1(t)$, which is also synchronized with $e(t)$. Given the width of $e(t)$ and the time order of $e(t)$ and $C_1(t)$ in synchronization mode, $e(t)$ always goes high when $C_1(t)$ is high. However, this is not guaranteed at the start of the communication. A rising edge of $e(t)$ when $C_1(t)$ is low would cause $C_1(t)$ to have a low pulse whose width may be too small for correct operation of the system. The *pulse width control* module in Fig. 5.6 ensures that a low value of $C_1(t)$ is never influenced by $e(t)$ and the duty cycle of the synchronized clock signal never exceeds 75 %.

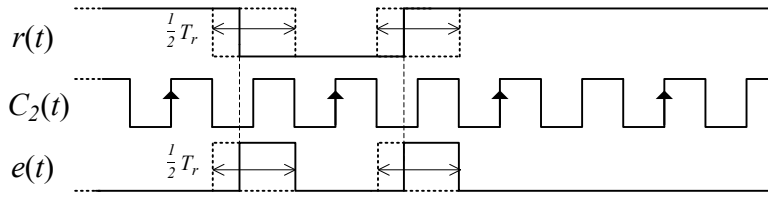


Figure 5.7: Edge detection signal $e(t)$ generated from $r(t)$ and $C_2(t)$ signals.

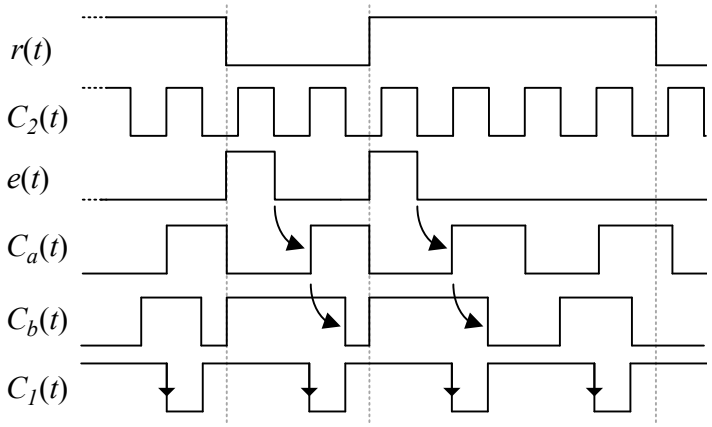


Figure 5.8: $C_a(t)$, $C_b(t)$ and $C_1(t)$ signals according on the phase difference between incoming signal $r(t)$ and local clock $C_2(t)$.

Figure 5.8 shows that the time difference between the falling edges of $e(t)$ and $C_1(t)$ is always $\frac{1}{4}T_r$. On the other hand, the width of the high pulses of $e(t)$ is variable between 0 and $\frac{1}{2}T_r$. Therefore, the time difference between $r(t)$ and $C_1(t)$ obeys to the condition

$$r(t)\updownarrow + \frac{T_r}{4} < C_1(t)\downarrow < r(t)\updownarrow + \frac{T_r}{4} + \frac{T_r}{2} \tag{5.24}$$

where $r(t)\updownarrow$ is any instant at which $r(t)$ makes a transition and $C_1(t)\downarrow$ is the time for the next falling edge of $C_1(t)$. Assuming that communication starts at $t = 0$ then $r(t)\updownarrow = kT_s$, where k is an integer. Considering that $T_s = T_r$, i.e., that the oscillator frequencies at sender and receiver are nominally equal. With these conditions, Eq. (5.24) will be

$$kT_r + \frac{T_r}{4} < C_1(t)\downarrow < kT_r + \frac{3}{4}T_r \tag{5.25}$$

Signal $C_1(t)$ is periodic and its falling edges occur in the range defined by Eq. (5.25), from which it follows that the phase difference between $C_1(t)$ and $r(t)$ obeys to the condition

$$\frac{-\pi}{2} < \Delta\varphi(t) < \frac{\pi}{2} \tag{5.26}$$

which is the same as the one given by Eq. (5.8). Therefore, the circuit satisfies the synchronization conditions and $C_1(t)$ (and $C_r(t)$) is a synchronized receiver clock.

Because of the internal delays of the synchronization circuit, changes in $r(t)$ or $C_2(t)$ do not affect $C_r(t)$ immediately, i.e., the delays shift the sampling range of the receiver. This effect is small, but in high speed systems (in which the delay may be comparable to the data rate period) it should be compensated by delaying $r(t)$. This is the purpose of module *Delay* in Fig. 5.6. It should be noted that the propagation delay is independent from the data rate or clock frequency. Therefore, the delay introduced by *Delay* is fixed and only depends on the details of the specific physical implementation. The impact of this delay and the effect of jitter are evaluated experimentally in the next section.

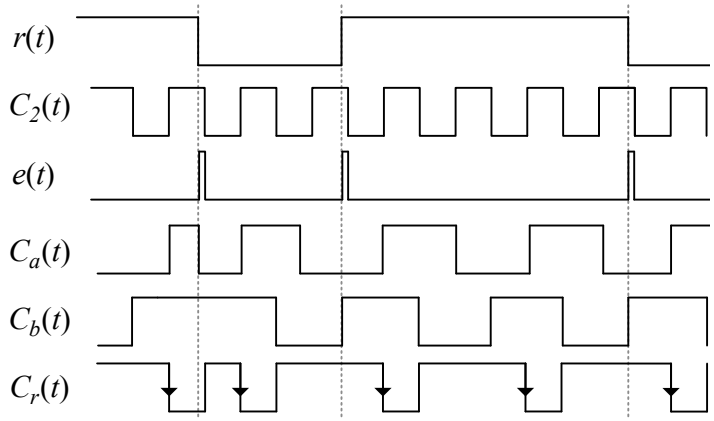


Figure 5.9: Shifting the negative pulses of $C_r(t)$ to the left (clock at sender is faster than clock at the receiver).

The phase difference between $r(t)$ and $C_2(t)$ is non-deterministic, so three different situations may arise, as analyzed next.

No sampling point adjustment. Figure 5.8 shows the signals for the case when $C_r(t)$ is in the correct range before a transition occurs in the incoming signal: the edges of $r(t)$ are more than $\frac{1}{4}T_r$ apart from the sampling edges, so there is no need to shift the pulses of $C_r(t)$. Signal $e(t)$ influences the shape of $C_a(t)$ and $C_b(t)$ as shown with dashed lines, but the resulting $C_r(t)$ signal is not affected.

Faster clock at sender. Figure 5.9 shows the case when $C_r(t)$ is adjusted because an edge of $r(t)$ occurs closely after the sampling edge of $C_r(t)$ (indicated by dashed lines). This case occurs when the receiver clock frequency is less than the transmitter's ($T_s < T_r$). The pulses of $C_r(t)$ are shifted to the left (advanced in time) by $\frac{1}{2}T_r$, so that the next sampling edge is in the synchronization range.

Shifting $C_r(t)$ to the left is only possible if the duty cycle is bigger than 50 %. Otherwise, shifting would cause the pulses to overlap and the receiver would lose one data bit. By using a sampling clock with 75 % duty cycle, overlapping never occurs.

Slower clock at sender. The third case is depicted in Fig. 5.10. This case the receiver clock frequency is higher than the transmitter's ($T_s > T_r$), so the circuit shifts the negative pulses of $C_r(t)$ by $\frac{1}{2}T_r$ to the right, delaying the next pulse.

When communication starts, there is no input signal for a while, so there is no synchronization between transmitter and receiver. Therefore, data must be preceded by a preamble (start bits) to synchronize the receiver. For this architecture, just one transition in the input is enough to synchronize the clock $C_r(t)$. Such a fast settling time is possible because an open-loop architecture is used.

The fact that $C_r(t)$ is synchronized with both the incoming signal and the system clock provides the ability for simultaneous communication with several SNs, a basic need for multitasking operation. For instance, each SN of the prototype IC has four independent CDR circuits. Each recovered clock is in synchronization with the received signal (to be in the correct sampling range),

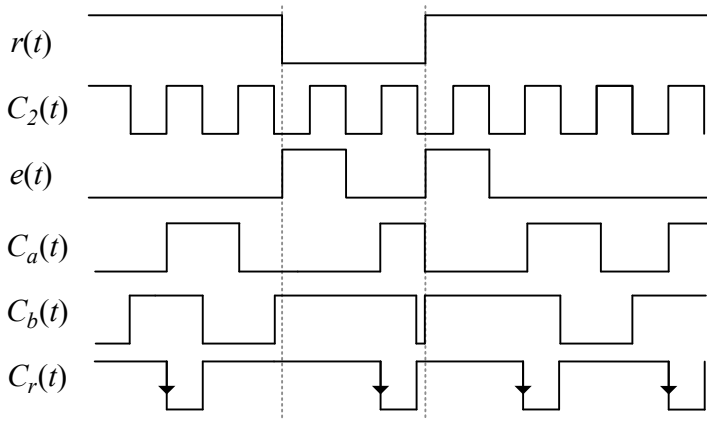


Figure 5.10: Shifting the negative pulses $C_r(t)$ to the right (clock at sender is slower than clock at the receiver).

and its edges are also inherently in synchronization with the system clock of the receiver. This enables each SN to handle communication with several independent SNs without buffering.

The gate-level circuit of Fig. 5.11 implements the block diagram of Fig. 5.6. Flip-flop FF1 and gate XOR1 implement the edge detector and creates $e(t)$. The clock dividers that produce $C_a(t)$ and $C_b(t)$ are implemented by FF2 (with INV1) and by FF3, respectively. An RS flip-flop (active low) is used for pulse width control with $C_2(t)$ and $C_1(t)$ connected to the set and reset inputs of the flip-flop, respectively. The output of the flip-flop ($C_r(t)$ signal) will be low when $C_1(t)$ is low and $C_2(t)$ is high. Regardless of the value of $C_1(t)$, $C_r(t)$ will remain low until $C_2(t)$ changes to low. The gray area is a NRZI decoder and is not a part of the synchronization circuit. For NRZ line encoding, the output of FF5 is used directly and gates XOR2 and FF6 are not necessary.

The synchronization circuit consists of eight logic gates in total, so it is small enough to have four instances included in the communication ASIC for the aforementioned wearable system. The circuit occupies 0.0022 mm^2 in a 0.35 µm CMOS process, which is much less than the area of the circuits listed in [CWJ11], the smallest of which occupies 0.1326 mm^2 in a 0.18 µm technology.

5.1.4 Experimental Results

This section describes the experimental results obtained with the circuit shown in Fig. 5.11, which corresponds the block diagram of Fig. 5.6. The circuit was implemented both on a low-power IGLOO FPGA [Mic13] and as part of a communications IC fabricated in 0.35 µm CMOS technology for use in a wearable sensor system. The results reported here were obtained with the IC version as is shown in Fig. 6.19 together with the sensor board Fig. 6.18. For the measurements, two sensors, SN1 and SN2, were connected to each other and set to communicate normally over a one-wire bidirectional link. The clock frequency of both sensor nodes is 16.383 MHz. All results shown in Figs. 5.12 to 5.17 were obtained with a digital oscilloscope. In all cases discussed next, SN1 is the sender and SN2 is the receiver.

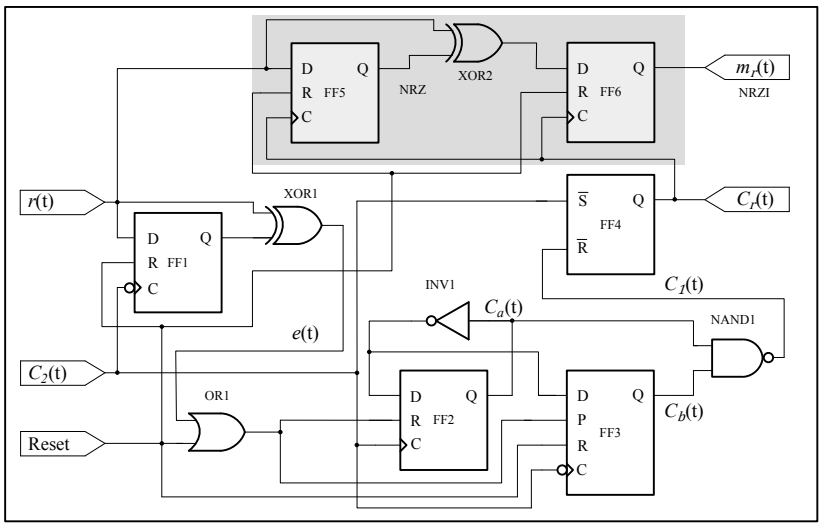


Figure 5.11: Combined circuit for synchronization and NRZI decoding.

In our prototype, each SN has a microcontroller with a crystal quartz oscillator. The clock pin of the IC is connected to the clock output of the microcontroller (without using any additional clock generator). Many microcontrollers also have their own accurate internal RC oscillator. The accuracy of these RC oscillators is less than the accuracy of crystal oscillators, but the prototype IC is able to work with both of them. In this case, no quartz crystal oscillator would be needed, as long as the total jitter stays below the threshold given by Eq. (5.13). The stability of the oscillator also affects the choice of line coding.

5.1.4.1 Normal operation

The oscillators of the SNs are running free and generate clock signals with a frequency that is randomly distributed around the nominal value. The difference between sender and receiver clocks cause the phase between the clocks to change over the time as shown in Fig. 5.12. This figure was captured using a 5 s time persistence; the last sample is indicated by a darker line. Figure 5.12 confirms that the synchronization circuit is keeping the negative edge of $C_r(t)$ in the sampling range, which is 61 ns for this setup.

This figure also shows that the delay between the incoming signal and $C_r(t)$ is about 7.5 ns. Signal $r(t)$ is captured at the input of the IC and $C_r(t)$ at the output (both in SN2). Therefore, the measured delay d includes the latency inside the IC, which is mainly due to the input and output pads:

$$d \approx d_{ipad} + d_{circuit} + d_{opad}, \tag{5.27}$$

where d_{ipad} is the delay of the input pad to the core, d_{opad} is the delay from the core to the output pad, and $d_{circuit}$ is the delay due to signal propagation in the synchronization circuit. Datasheet information for the pad cells indicates that $d_{ipad} + d_{opad} \approx 5$ ns, so that $d_{circuit} \approx 2.5$ ns.

The delay $d_{circuit}$ causes a time skew of sampling point, because it decreases the safe zone where the jitter can occur without problems and increase the BER. However, in comparison with

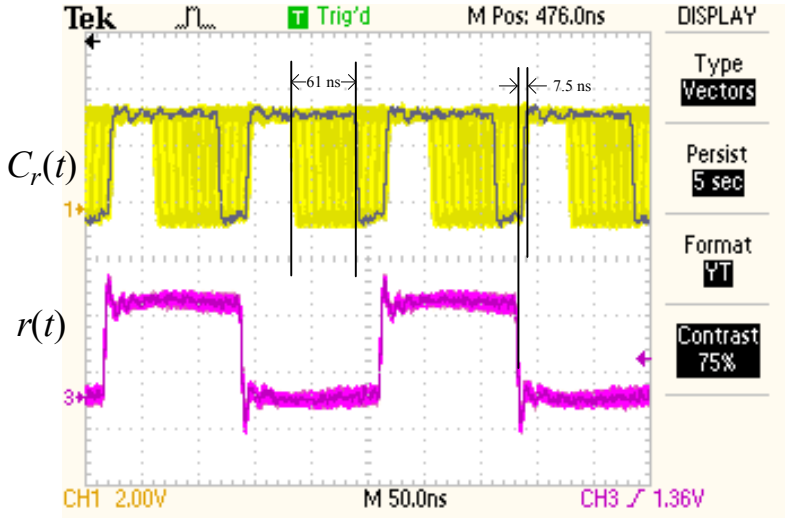


Figure 5.12: Signals captured using 5 s persistence: $r(t)$ is the incoming signal on the wire and $C_r(t)$ is the local clock signal. (The edges shown directly to the right of the sampling range are raising edges.)

the data rate period (61 ns), $d_{circuit}$ is small and its effect on synchronization is typically negligible. If necessary, the effect of $d_{circuit}$ is deterministic and can be compensated by delaying $r(t)$ (using module *Delay* of Fig. 5.6).

The following experimental results show the operation of the synchronization circuit according to the phase difference between the $r(t)$ and $C_2(t)$ signals, as discussed in Section 5.1.3.

No sampling point adjustment. Normal operation of the circuit is depicted in Fig. 5.13, which shows signals $r(t)$, $C_r(t)$ and $m_r(t)$ during data transmission over a copper wire with a data rate of 8.1 Mbps. Signal $C_r(t)$ is the synchronized clock signal in SN2 and $m_r(t)$ is the received message after NRZI decoding. In this case, $C_r(t)$ is a periodic signal, since it is always in the correct sampling range. Samples of $r(t)$ are taken at the negative edges of $C_r(t)$ and the decoded signal appears at the output almost immediately. Other parts of the receiver would typically use $m_r(t)$ at the positive edge of $C_r(t)$ for further processing.

Slower clock at sender. Figure 5.14 illustrates circuit operation when the clock frequency of SN2 is higher than the frequency of SN1. The circuit modifies $C_2(t)$ to be in the correct sampling range by shifting the signal one $\frac{1}{2}T_r$ to the right. The gray pulse of signal $C_r(t)$ in Fig. 5.14 was inserted to highlight the place where a pulse would occur if no correction had been performed. As can be seen in the figure, the time difference between edge of $r(t)$ and the falling edge of $C_r(t)$ would be less than $\frac{1}{4}T_r$. So, the circuit shifts $C_r(t)$ to the right to be in correct sampling range. Afterwards, $C_r(t)$ repeats periodically until another correction is necessary.

Faster clock at sender. Figure 5.15 corresponds to the situation in which the clock frequency of SN2 is less than the frequency of SN1. As shown in the figure, the circuit shifts $C_r(t)$ one $\frac{1}{2}T_r$ to the left to be in the correct sampling range, whenever the time difference between the falling edge of $C_r(t)$ and the next edge of $r(t)$ is less than $\frac{1}{4}T_r$.

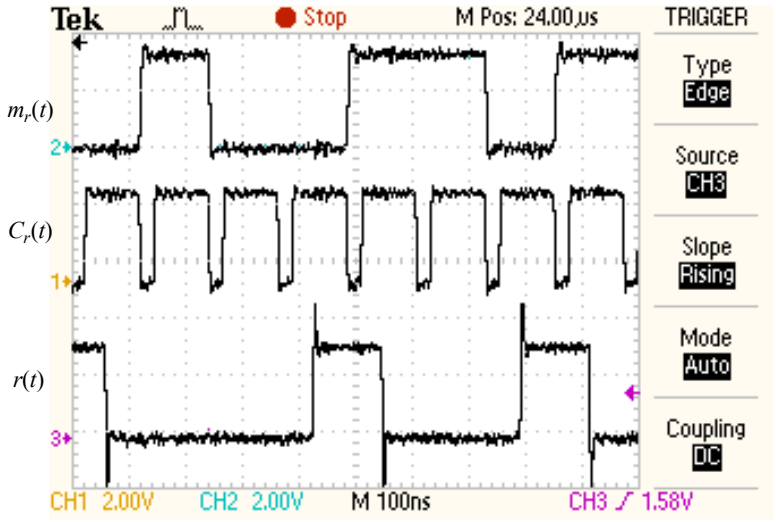


Figure 5.13: Signals at the receiver when no local clock adjustment is needed: data input $r(t)$, locally synchronized clock $C_r(t)$ and NRZI decoded data signal $m_r(t)$.

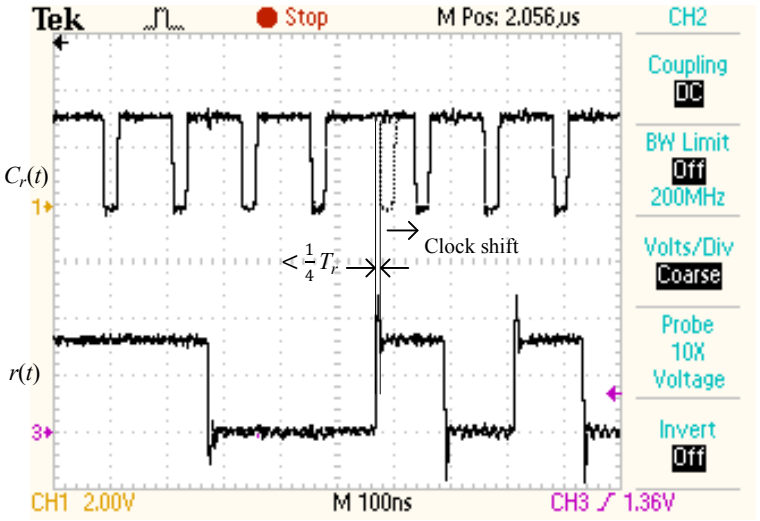


Figure 5.14: Correction of receiver clock $C_r(t)$ when $T_s > T_r$.

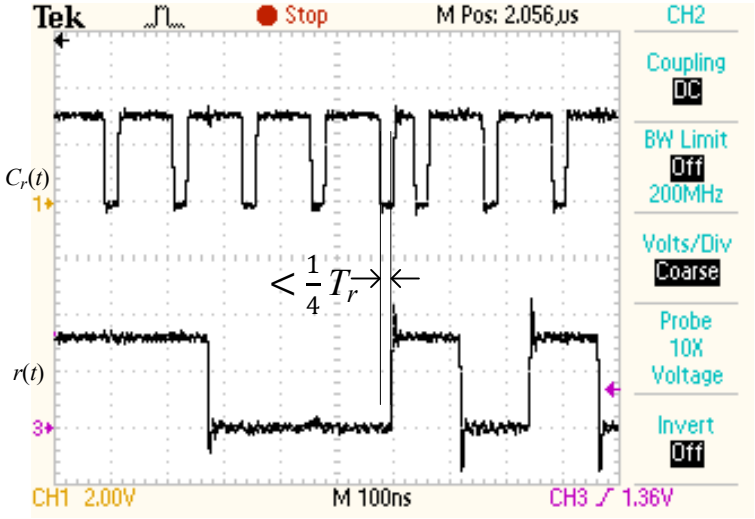


Figure 5.15: Correction of receiver clock $C_r(t)$ when $T_s < T_r$.

5.1.4.2 From Sleep Mode to Active Mode

This synchronization circuit is a part of a system that includes a mechanism to monitor communication activity and put the system in sleep mode when possible, so as to reduce energy consumption. In this situation, $C_r(t)$ remains high, as shown in Fig. 5.16. When a signal appears at the input, the system clock control module enables the clock and puts the system in active mode. The synchronization circuit correctly starts generating $C_r(t)$ as shown in Fig. 5.16: $C_r(t)$ falls at the correct time to sample the second incoming bit. The first bit is defined to be a start bit that is used to determine the start of communication; the following data bits are correctly received even immediately after waking up the system. Both in active or in sleep mode, one preamble bit is enough for the synchronization circuit to work properly.

5.1.4.3 Bit Error Rate

The BER was measured for three different types of interconnection: a) copper wire, b) single conductive yarn in relaxed mode, c) four parallel conductive yarns. All connections are 1 m long. A good equivalent circuit for a conductive yarn is composed by a resistor and an inductor in series [ZDD+12], whose values change as the yarn is stretched. For the conductive yarns used in the measurements, the parameters in relaxed mode are R=1.56 Ω/cm and L=8 nH/cm.

Figure 5.17 shows the eye diagram of the signals obtained when using four parallel conductive yarns for connecting the nodes, which is the normal situation in the target wearable system, since a single yarn is often too fragile to withstand the wear. The diagram was generated by sending randomly varying data ($2^{64} - 1$) and using the oscilloscope's infinite persistence feature. The diagram shows that the receiver clock is always in the right range of sampling.

Figure 5.18 shows the measured BER as a function of the unit interval (UI). The BER was calculated by using the dual-Dirac methodwhich determines BER as a function of Random Jitter (RJ) due to noise from different sources (with a zero-mean Gaussian distribution) and Deterministic

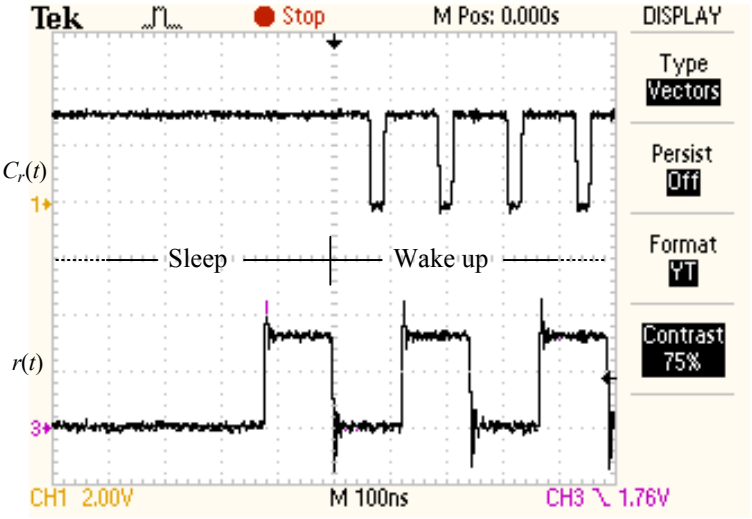


Figure 5.16: Initiating generation of $C_r(t)$ after exiting from sleep mode.

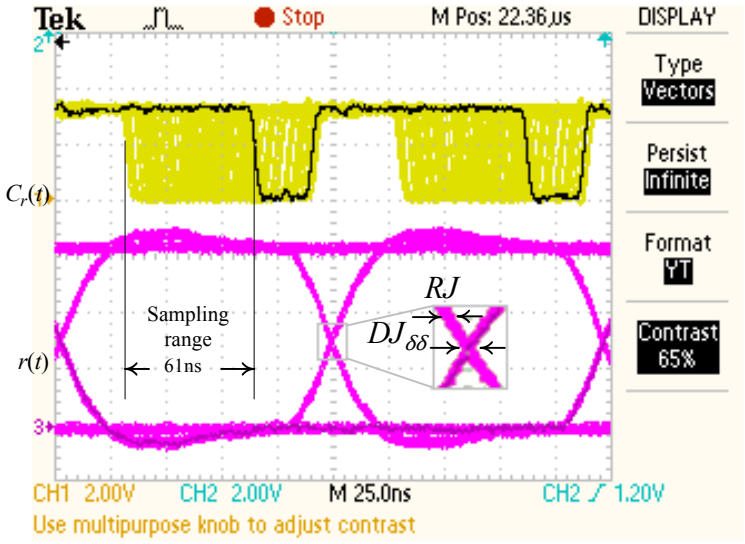


Figure 5.17: Eye diagram and synchronized clock signal $C_r(t)$ when using four parallel conductive yarns.

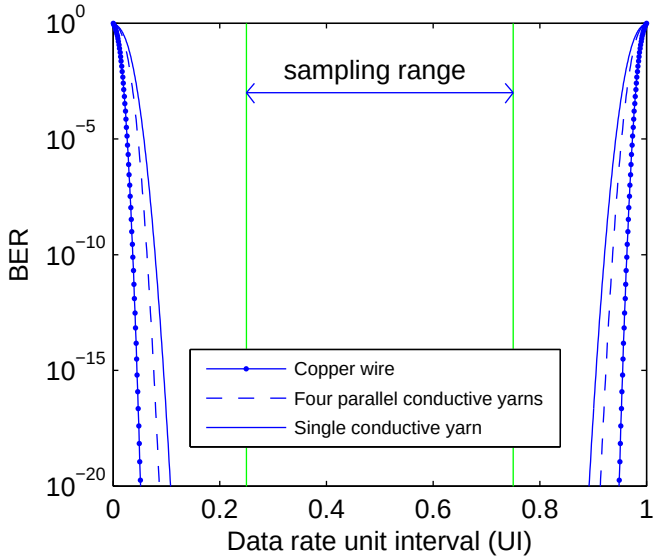


Figure 5.18: Measured BER and safe sampling range for three different interconnections.

Jitter (DJ) due to signal losses, duty-cycle distortion, crosstalk and other causes [Ste04]. Table 5.1 lists the measured peak-to-peak value of RJ and $DJ_{\delta\delta}$. As Figure 5.18 shows the copper wire has lower BER than conductive yarns; the probability of error increases with increasing the resistance and inductance of the yarn, because of the increasing signal attenuation. This figure also shows that if the clock synchronization circuit keeps the receiver clock in the sampling range, then the data will be detected correctly with high probability.

Random jitter is mainly due to additive white noise superposed on the received signal. In systems with low SNR, such as wireless communication, the random part may be significant and increase the total jitter noticeably. Therefore, in those systems the total jitter may be larger than the limit imposed by Eq. (5.13). The relation between BER and SNR depends on the communication method. In the absence of deterministic jitter, the conventional analysis of baseband binary signals shows that BER due to additive white Gaussian noise (AWGN) with variance σ^2 at the input is given by

$$BER = Q\left(\sqrt{\frac{S}{2N}}\right), \qquad S = \frac{A_0^2 + A_1^2}{2}, \qquad N = \sigma^2 \tag{5.28}$$

where Q denotes the error probability density function, A_0 and A_1 the signal levels, S the signal

Table 5.1: The values of jitters.

Conductor	**RJ (ns)**	**$DJ_{\delta\delta}$ (ns)**
Copper wire	1	1.5
Four parallel conductive yarns	1.6	2.5
Single conductive yarn	2	3

power, and N the noise power [Gib02]. However, deterministic jitter can be more significant than random jitter even if the SNR is high. The total BER is the result of both SNR (appears as random jitter) and deterministic jitter. Regardless of the communication method, the BER can be calculated by measuring the jitter values and using the Dual-Dirac method [Ste04].

5.1.4.4 Comparison with Other Synchronization Circuits

Many CDR circuits have been introduced in the last years. Table 5.2 summarily compares the present implementation to some other recent circuits. The maximum measured data rate for this work is 35 Mbps ($T_r = \frac{1}{35 \times 10^6\,\mathrm{Hz}} = 28.57\,\mathrm{ns}$). The settling time is one data period. The total jitter must be less than $\frac{1}{8}T_r$. Therefore, the peak-to-peak jitter must be less than $\frac{2}{8}T_r = 7.14\,\mathrm{ns}$.

Considering that the aim of the proposed CDR circuit is to minimize power consumption and area, a quick look at the table leads to the conclusion that this was readily achieved. Even though a 0.35 µm CMOS technology, with 3.3 V power supply, is used, the present work attains the smallest area and power consumption of all the works listed in the table. It is noticeable that the low value of power consumption is the consequence of a small-size circuit (fewer active components), but also due to a lower clock frequency operation, a factor that in general sets the average consumption during logic switching, which is the biggest slice of consumed energy. However, if a simple relative frequency scaling is applied to the power consumption in order to predict how much power the circuit would consume at a higher clock rate, the estimated power is still substantially smaller than any of the other circuit counterparts. Although this is a weak estimate to be utilized for a direct comparison, it does suggest that in fact the proposed CDR architecture is very power efficient. In addition, the settling time just takes a single period of the data-rate, which is also faster than any of the other closed-loop circuits in the table.

These characteristics are obtained at the cost of clock jitter, which is the highest present in the table. This is mainly due to the technique used for synchronization, which always adjusts the sample clock by $\pm\frac{1}{2}m$ of the data rate. Increasing m would increase the accuracy of sampling and the acceptable jitter range as well, but, for the same data rate, would require increasing the system clock frequency, which would also increase the power consumption. Nevertheless,as expected from analysis of the circuit and the results presented in this section, as long as signal jitter is less than 25 % of the data rate as happens with our target application, the circuit with $m = 2$ is able to correctly recover the data.

Table 5.2: Comparison with other works

CDR circuit	Open-loop	Process (μm)	Frequency (Mbps)	Clock jitter (pk-pk)(ps)	Settling time (bits / time(μs))	Power (mW)	Supply voltage (V)	Core area (mm^2)
This work	y	0.35	35	7.14×10^3	1 / 28.56×10^{-3}	0.051	3.3	0.0022
[KLLJ13]	y	0.065	100	242	N/A	0.36	1.2	0.07
[PCS+08]	y	0.18	180–720	N/A	1 / <5.56×10^{-3}	8.2	1.8	0.185
[CWJ11]	n	0.18	100	199.66	400 / 4	6.649	1.8	0.16
[CW09]	n	0.18	5120–6400	2.12	N/A	136	1.8	0.8
[LL08]	n	0.18	662–3125	62.2	>331000 / >500	60	1.8	0.1326
[WLS+06]	n	0.18	2500	88	4862 / 3.89	N/A	1.8	0.133
[YCH+06]	n	0.18	155.52–3125	467	>1552 / 100	95	1.8	0.88
[YCL06]	n	0.35	200–2000	120	>60000 / >30	170	3.3	0.4

5.1.5 Conclusion

A very small, fully digital open-loop clock and data recovery method for use in wearable systems has been described. The receiver clock comes from a local, autonomous oscillator that is running at a frequency of twice the data rate and synchronization is based on open-loop selection of the correct phase of the clock in receiver synchronously with the incoming signal.

The achieved performance represents a trade-off between closed-loop and open-loop methods. The relatively higher jitter of the recovered clock is compensated by several favorable characteristics. The circuit size is much smaller than previously reported implementations, because it only consists of 8 logic elements and occupies $0.0022\,\mathrm{mm}^2$ in a $0.35\,\mu\mathrm{m}$ CMOS process. The circuit requires only one transition in the input for the alignment of the receiver clock in the correct sampling rang. Therefore, it is faster than any closed-loop method. By increasing the system clock frequency, the synchronization performance of the circuit would improve, but the power consumption would also increase. The proposed circuit works with the minimum system clock frequency of twice of data rate. The experimental results obtained with a prototype ASIC show that the circuit can be successfully used in a wearable sensor system, as in this case the peak-to-peak value of the jitter does not affect the sampling range and the circuit recovers the data with high BER.

5.2 Time Synchronization

This section presents a one-way method for synchronization at the MAC layer of nodes and a circuit based on that in a wearable sensor network. The proposed approach minimizes the time skew with an accuracy of $\frac{1}{2}$ clock cycle on average. The work is aimed at to use in the IC as described in Chapter 6. In particular, we address the need for good time synchronization in the simultaneous acquisition of surface electromyographic signals of several muscles. Inertial sensors are also present in order to provide 3D information about the movements of the limbs. In our main application case, the electrodes are embedded in patient clothes connected to SNs equipped with ADC converters. The SNs are connected together in a network using conducting yarns embedded in the clothes.

In the context of such wearable sensor networks, the main contributions of this work are the evaluation of existing protocols for synchronization, the description of a simpler, resource-efficient synchronization protocol, and its analysis, including the determination of the average local and global clock skew and of the synchronization probability in the presence of link failures.

5.2.1 Motivation

The development of the synchronization protocol described here was motivated by the requirements of the wearable system presented in previous chapters. Each sensor node is capable of acquiring sEMG signals from electrode pairs and kinematic data from 3D inertial sensors.

Since the relative timing of the acquired data is relevant for subsequent processing, a synchronization mechanism for time stamping of data is required. Data rates in BAN applications

are significantly higher than in other sensor networks (e.g., 32 kbps for a 16-bit EMG sensor) [LBM+11]. Therefore, the synchronization protocol has to ensure timing accuracy in the range of a few microseconds in a wired, multi-hop network. The main aspects that have been considered in the design of the time synchronization protocol are:

- **Energy efficiency:** Due to the portability requirements of the aforementioned system, its sensor nodes use a small battery for power supply. To ensure an adequate operation time with such a limited energy source, the efficient usage of energy in all parts of the system has to be considered. The communication infrastructure, including transmitter and receiver, consumes a significant amount of energy when they are in active mode. Therefore, the number of control messages and transmitted data packets has to be reduced as much as possible. In the proposed protocol, synchronization is based on one-way message exchange, which consumes less energy than two-way methods. Utilizing a one-way method may increase the processing cost or the time to achieve synchronization, but the performance of this work is not affected by aforementioned disadvantages.

- **Accuracy:** Unknown delay and latency of messages generally affect on time synchronization. Messages experience different delays while passing between network layers. Increasing the number of involved layers produces a cumulative increase of end-to-end delay. This applies specially to the network layer, because the buffering of messages and the variable service time due to changing network traffic make it hard to estimate message delay and reduce the accuracy. Therefore, many synchronization protocols have been implemented in the MAC layer, so as to achieve high accuracy and fast synchronization. The current protocol has also been designed for MAC layer use. s

- **Simplicity:** The limited computational resources available in sensor nodes always constitute a key factor, which has to be considered in the design of both hardware and software. Keeping the communication and data processing delays within a fixed range reduces message processing and the size of the hardware needed to implement the protocol in the MAC layer. In comparison to two-way protocols, because of minimizing delay changes, the proposed method requires less calculations.

A one-way master-slave sender-to-receiver mechanism that satisfies these criteria, while enabling a resource-efficient hardware implementation, is described in the next section.

5.2.2 Description of the Synchronization Protocol

This section describes the proposed time synchronization protocol and its intended deployment. Figure 5.19 shows an example of a mesh network of SNs connected to each other by wires (conductive yarns in the intended application). All SNs are equipped with a multi-port network module. Such a network module includes synchronization and packet routing sub-modules working independently of each other. Each device can perform time synchronize with the sender node over one

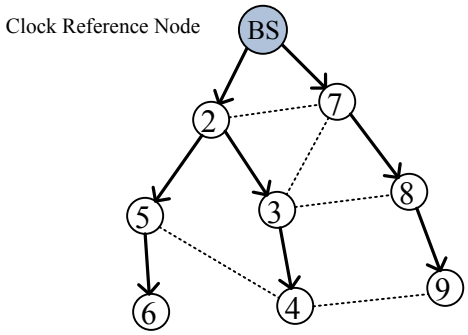


Figure 5.19: A network of sensors with the BS node acting as clock time reference. The bold lines indicate the tree used for synchronization messages.

port, while participating in packet routing with the other free ports. Node 1 is the BS responsible for network management, master clock time reference and data collection. A routing protocol is needed to route packets from SNs to BS, and setting, timing and control messages from BS to SNs. Here the energy-efficient routing protocol from [DFT10] is used, which is based on the combination of source routing and minimum cost forwarding.

Before starting normal activity, the system must go through a setup phase. In this step, the routing protocol finds all minimum-cost paths between BS and SNs, as shown by bold lines in Fig. 5.19. The dashed lines are redundancy links that increase the robustness of the system against line breaks caused by wear. Recall that the SRMCF protocol requires that each node determines its neighboring node on the path with minimum cost to the BS (called the *near-node*), and then sends the path information to the BS, who keeps a database of available SNs and the corresponding paths. Synchronization information is transmitted as control messages over the minimum-cost spanning tree built in this phase.

After the network setup, the BS as clock reference synchronizes the network by sending timing information whenever a neighbor SN sends a request. Each SN synchronizes with its *near-node* and rejects timing messages from the other nodes. For instance, node 3 will synchronize with node 2 and reject messages from nodes 7 and 8. Because messages received over different paths experience different delay variations (due to different number of hops, network traffic and packet loss), accepting timing information from various sources would cause message jitter.

End-to-end communication time is usually not deterministic and suffers specially from buffering in the network layer. This protocol is designed for implementation in the MAC layer in order to reduce the effects of delay and the clock time skew. A dedicated hardware transmitter module on each node is responsible for the generation of the timing messages. Also it should be noted, as referred earlier that to reduce energy consumption, the communication modules are in sleep mode when there is no data to be transfered.

Figure 5.20 shows a data frame with a timing message, both as constructed by the sender and as processed by the receiver. The preamble of a timing message is a string of 1s, which is used to wake up the receiver before starting the actual data transfer. The MAC header enables the receiver to recognize an incoming timing message. Then the receiver activates a hardware

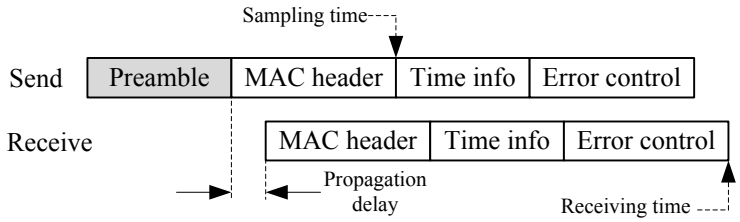


Figure 5.20: Format of the timing information message (MAC layer) and received message at the receiver.

module designed to process the timing information directly without involving of the other parts of the receiver. Immediately after sending the MAC header, the master samples the current time, formats the timing information, and sends it. Sampling the time at the sender in this way reduces the delay error due to the message sending process. Error control information is put at the end of the message, so that the message integrity can be checked by the receiver.

Delay due to propagation of signals over the link between the nodes is typically very small and negligible (for instance, about 3.3 ns per meter for our wearable network). The main delay is due to the difference between the instant when time is sampled at the sender and the message is received. For the data frame shown in Fig. 5.20, the transfer takes a fixed number of system clock cycles, because the number of bits inserted in the message after the sampling of time is fixed. Therefore, the receiver always gets messages after a fixed delay (ignoring clock drift and propagation time). If the transmitter adds this fixed number of clocks to the sampled time, then the receiver has all the necessary information to determine exactly the instant when the incoming packet arrived (as measured by the clock of the sender). If the receiver validates the new timing information, then it can set its own time to the received time without further processing.

After synchronizing its time with sender node, each SN sends a timing message to its successors on the clock time tree. For example in Fig. 5.19, node 2 sends timing messages to the nodes 3 and 5. After adjusting their time, nodes 5 will send timing messages to nodes 5 and 4, and node 3 will send a message to node 4.

We assume that each SN uses the time information to manage a globally synchronized clock signal *Clk-sync*, whose frequency is smaller than the system frequency. A simple way to generate such a clock signal is to use an auto-reload down-counter: when the value of the counter reaches zero, an active transition of *Clk-sync* is generated and the counter is reloaded with a predefined value (equal for all nodes).

In this case, synchronizing *Clk-sync* signals is equivalent to keeping the values of all counters synchronized. The counter in each node is driven by the local system clock, which exhibits some clock skew relatively to the system clock of other nodes. To limit the skew between nodes, each SN periodically updates its counter with the value specified in the timing message coming from its reference node.

A SN may also manage a time stamp to annotate any acquired data. We assume that the current time stamp at each SN is determined by an up-counter driven by the globally synchronized

Clk-Sync signal. The time resolution depends on the frequency of *Clk-Sync*, which is application dependent and must be selected according to characteristics of the sensed phenomenon. For example, the sampling rate of electromyography signals is 1 kHz to 2 kHz, which much smaller than the system clock frequency (usually above 1 MHz).

The use of *Clk-sync* ensures that the time stamp counter operates at the same frequency in all nodes. However, it is still necessary to ensure that its contents are the same by using a dedicated message. After starting up, each SN first synchronizes the *Clk-sync* down counter. Then, the time stamp counter contents is synchronized once. Afterwards, the *Clock-sync* counter must be synchronized periodically to compensate for the drift of the local system clock.

Under the proposed protocol, the processing of timing messages at the transmitter side is limited to adding a fixed number to the sampled time, and the receiver does no processing to estimate clock time skew. It is important to reduce any processing as much as possible, because the protocol is intended for hardware implementation in an energy-constrained environment.

Elimination of the effects of sending and receiving delays on timing message as described above minimizes the clock time skew between SNs and leads to a high precision synchronization mechanism for wired wearable networks with a simple hardware implementation.

5.2.3 Analytic Characterization of the Protocol

This section provides an analytic characterization of the proposed protocol's behavior for use in wired wearable sensor networks. A comparison with PTP is done where appropriate. The aspects addressed are: delay and offset bounds, average local and global time skew, and probability of synchronization as a function of packet loss.

5.2.3.1 Instantaneous Delay and Skew

We assume that all nodes have the same nominal system clock period τ. However, the actual clock period of node i will exhibit a small difference due to random local clock drift p_i, so that in general the clock period τ_i at node i is given by:

$$\tau_i = \frac{1}{f_i} = \frac{\tau}{1+p_i} \tag{5.29}$$

The p_i depends on the oscillator that it can be in range of few ppm for a stabilized oscillator with crystal quartz. Consider the multi-hop network of Fig. 5.19 with the minimum cost paths forming a spanning tree, and assume that node 1, the BS, is the time reference, which periodically broadcasts its time to other nodes. Figure 5.21 a) shows the synchronization timing diagram for PTP. The time reference node (node 1) starts by sending a message to the client node (node 2) at time $T_M(t_1)$ (master clock time). Node 2 receives it at $T_S(t_2)$ (slave clock time) and replies at $T_S(t_3)$ with a message that is received by node 1 at $T_M(t_4)$. Node 1 calculates the delay and offset

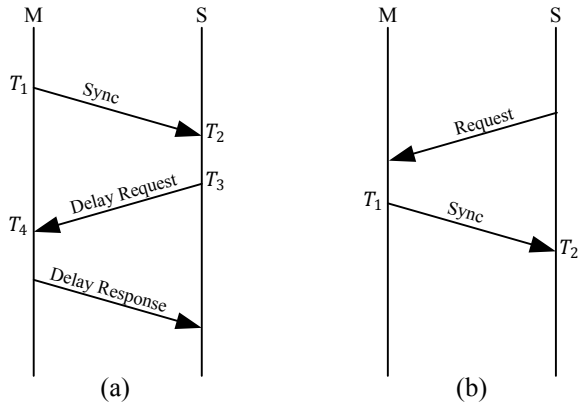


Figure 5.21: Timing diagram: a) PTP protocol, b) proposed protocol

according to Eqs. (5.30) and (5.31) [LE02, IEE08b]:

$$\text{delay} = \frac{(T_S(t_2) - T_M(t_1)) + (T_M(t_4) - T_S(t_3))}{2} \tag{5.30}$$

$$\text{offset} = \frac{(T_S(t_2) - T_M(t1)) - (T_M(t_4) - T_S(t_3))}{2} \tag{5.31}$$

If the time value included in the *Sync* request is $T_M(t_1)$, then $T_S(t_2)$ is given by

$$T_S(t_2) = T_M(t_1) + s\,\tau_1 + c\,\tau_d + d_\ell + r\,\tau_2 + S_r(t), \tag{5.32}$$

where s is the number of clock cycles used in internal processing after $T_M(t_1)$, c is the number of bits transmitted at data rate period τ_d, d_ℓ is the propagation delay between nodes 1 and 2 (the sum of delays of line and I/O ports) , r is the number of clock cycles for message reception at the receiver (with clock period τ_2), and $S_r(t)$ determines the sampling range and is the time difference between signal appearance at the input of the sampling module of receiver and the actual sampling instant. The data rate is not necessarily equal to the system clock frequency of the transmitter, so that in general we have

$$\tau_d = m\,\tau_1, \qquad m \in \mathbb{N}^+. \tag{5.33}$$

The propagation delay $d_{\ell_{1,2}}$ is usually very small in comparison with the clock speed. We will also assume that $d_{\ell_{1,2}} = d_{\ell_{2,1}} = d_\ell$. Parameters c and r are usually fixed, depending on the packet size. Because we assume that the implementation is in the MAC layer, the receiver node processes the message almost immediately after receiving it, i.e., without putting it in a queue,

which decreases the uncertainty delay at the receiver. Under these conditions we have:

$$\begin{aligned} \text{delay} &= \frac{((s+mc)\tau_1 + d_\ell + r\tau_2 + S_r(t))}{2} + \\ &\quad \frac{((s+mc)\tau_2 + d_\ell + r\tau_1 + S_r(t))}{2} \\ &= d_\ell + S_r(t) + \\ &\quad \frac{(s+mc+r)\tau}{2}\left(\frac{2+p_1+p_2}{(1+p_1)(1+p_2)}\right). \end{aligned} \tag{5.34}$$

Under the same assumptions, the offset is given by:

$$\text{offset} = (s+mc+r)\tau\left(\frac{p_2-p_1}{(1+p_1)(1+p_2)}\right). \tag{5.35}$$

Since p_1 and p_2 are very small values, Eq. (5.34) simplifies to

$$\text{delay} \approx d_\ell + S_r(t) + (s+mc+r)\tau. \tag{5.36}$$

Since PTP measures time only in multiple units of *tau* and, for this application, $d_\ell < \tau$ and $S_r(t) < \tau$, calculation of the delay with a round trip message exchange would always produce the constant value $(s+mc+r)\tau$. If there is always a fixed transmission delay, then there is no need to recalculate it. On the other hand, the offset value calculated by Eq. (5.35) is very small and returns zero value by a processor or microcontroller. In other words, there is no need to have a round trip message, when the delay is fixed. In this case, a single message (as in Fig. 5.21(b)) is enough to send timing information, while achieving the same precision. Therefore a simpler protocol with fewer message exchanges can be used. In the remainder of this subsection, the one-way method used by the proposed approach to achieve minimum clock time skew is analyzed. Here the clock time skew refers to the total time deviation of each node from the time reference.

Although SNs communicate in asynchronous mode, they are implemented as synchronous logic circuits, where nodes process data at the edge of the clock signal. Since the hardware clock generators of each node are independent, two nodes do not have exactly the same clock signal, even without any error or failure. The situation is illustrated by the timing diagram of Fig. 5.22, which shows the events at two communicating nodes. The transmitter sends data at the positive edge of the clock and the receiver detects incoming data at the negative edge. In this example, bit b_{n-1} is sent at time t_1, and the receiver will acquire it after $d_\ell + S_r$. The ideal value of $S_r(t)$ is half of the data rate period, so that sampling occurs exactly in the middle of the incoming signal. In practice, due to receiver clock variation and jitter, the sampling point changes around the middle of incoming signal, as shown with gray area in Fig. 5.22.

Equation (5.32) still applies for a single message (Fig. 5.21 b). The proposed approach determines that the transmitter add n clock cycles to T_M in order to minimize the time skew or error.

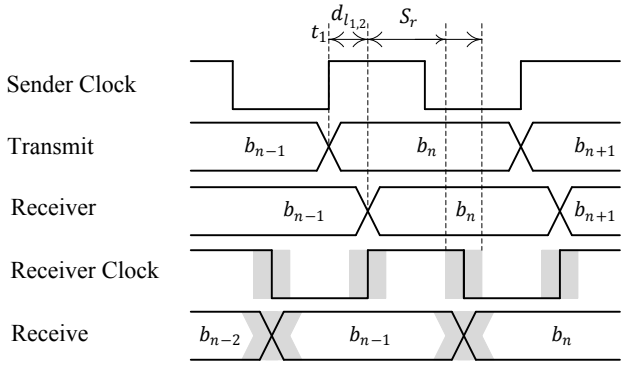


Figure 5.22: Clocks and signals in sender and receiver nodes

Assuming $\tau_1 \approx \tau_2$, we have

$$T_S(t_2) \approx T_M(t_1) + n\tau_1, \quad n \in \mathbb{N} \tag{5.37}$$

The integer value of n calculated from Eqs. (5.32) and (5.37) is

$$\begin{aligned} n &= \left\lceil \frac{(s+mc)\tau_1 + d_\ell + r\tau_2 + S_r(t)}{\tau_1} \right\rceil \\ &\approx (s+mc+r) + \left\lceil \frac{d_\ell + S_r(t)}{\tau_1} \right\rceil. \end{aligned} \tag{5.38}$$

By adding n to the time included in the packet, the receiver sets its own time to be approximately equal to the time of the reference node:

$$T_S(t_2) \approx T_M(t_2). \tag{5.39}$$

Since t_1 and t_2 are arbitrary times, this approach ensures that Eq. (5.39) is always valid.

Using Eqs. (5.32) and (5.37), the instantaneous time skew between transmitter and receiver is found to be:

$$\begin{aligned} C_S(t) &= T_M(t) - T_S(t) \\ &= (s+mc)\tau_1 + d_\ell + r\tau_2 + S_r(t) - n\tau_1 \\ &= r(\tau_2 - \tau_1) + d_\ell + S_r(t) - \tau_1 \left\lceil \frac{d_\ell + S_r(t)}{\tau_1} \right\rceil. \end{aligned} \tag{5.40}$$

In comparison with other terms, the value of $(\tau_2 - \tau_1)$ is very small (e.g. 30 ps at 20 MHz with a crystal quartz oscillator) and may be ignored. Therefore, time skew mainly depends on propagation time and sampling range $S_r(t)$.

This previous analysis applies to all nodes that are *one hop* away from the reference node but it should be noted that regardless of the master clock variation or position, the slave node always follow the master clock. So, the above calculation can be applied to all nodes with one hop distance from each other. In this way, the time skew accumulates as the hop count increases. In other words, for a node which is h hops away from the time reference node, time skew under the

proposed approach is given by:

$$C_S^h(t) = h\left(d_\ell + S_r(t) - \tau\left[\frac{d_\ell + S_r(t)}{\tau}\right]\right). \tag{5.41}$$

where the upper index of C_S^h refers to the number of hops.

The variation of the time skew can be estimated from Eq. (5.41) as

$$\frac{\partial C_S^h}{\partial t} = h\left(\frac{\partial d_\ell}{\partial t} + \frac{\partial S_r}{\partial t}\right). \tag{5.42}$$

Signal propagation is almost constant in a wired wearable network, so the previous expression simplifies to

$$\frac{\partial C_S^h}{\partial t} \approx h\frac{\partial S_r}{\partial t}. \tag{5.43}$$

This means that the time-skew deviation is a linear function with $S_r(t)$. So, it is expected that the any variation on the recovered clock at the receiver appeases linearly on the synchronized time. Here no restriction is made on $S_r(t)$ and it can be any arbitrary synchronization method.

5.2.3.2 Average Time Skew

The value of n given by Eq. (5.38) depends mostly on constant parameters, but varies with the instantaneous value $S_r(t)$. The use of a fixed value of n implies that $S_r(t)$ has to be constrained to a certain range. For baseband communications the optimal average value of $S_r(t)$ is the middle of the incoming data bit. This condition can be used to calculate a fixed value of n for use in the proposed approach. From Eq. (5.33) we have

$$\langle S_r(t)\rangle = \frac{\tau_d}{2} = \frac{m}{2}\tau, \tag{5.44}$$

where $\langle S_r\rangle$ is the average of $S_r(t)$. Using $\langle S_r\rangle$ in Eq. (5.38) results in

$$n = (s + mc + r) + \left[\frac{d_\ell + \frac{m}{2}\tau}{\tau}\right]. \tag{5.45}$$

Since $d_\ell \ll \tau$ in wired wearable sensor networks, Eq. (5.45) simplifies to

$$n = (s + mc + r) + \left\lceil\frac{m}{2}\right\rceil \tag{5.46}$$

and the average skew is

$$\langle C_S^h\rangle = h\left(d_\ell + \frac{m}{2}\tau - \left\lceil\frac{m}{2}\right\rceil\tau\right). \tag{5.47}$$

This equation gives different results for even and odd values of m:

$$\langle C_S^h\rangle = \begin{cases} h\left(d_\ell + \frac{1}{2}\tau\right), & m \text{ odd} \\ \\ hd_\ell, & m \text{ even} \end{cases} \tag{5.48}$$

For even m, $\langle C_S^h \rangle$ grows as the number of hops and only depends on the signal propagation delay between the nodes; when m is odd, $\langle C_S^h \rangle$ also depends on the system clock period.

The behavior of $\langle C_S^h \rangle$ for odd values of m can be controlled by a modification of the adding strategy. For such a system, the average time skew at nodes one hop away from the reference node is

$$\langle C_S^1 \rangle = d_\ell + \frac{1}{2}\tau. \tag{5.49}$$

Now for the second hop, add the value $n-1$ instead of n to the reference time in the message. The average time skew for nodes that are two hops away becomes

$$\langle C_S^2 \rangle = d_\ell + \frac{1}{2}\tau + d_\ell - \frac{1}{2}\tau = 2d_\ell. \tag{5.50}$$

As a consequence, the average time skew decreases and the value becomes the same as the one for even m. By continuing this procedure of adding n for nodes with odd hop counts and $n-1$ for nodes with even hop counts, $\langle C_S^h \rangle$ for nodes h hops away from the the reference is

$$\langle C_S^h \rangle = d_\ell \sum_{i=1}^{h} i + \sum_{i=1}^{h} \frac{1}{2}\tau(-1)^{1+i}, \quad m \text{ odd}. \tag{5.51}$$

In general, for any $m \in \mathbb{N}^+$, Eqs. (5.48) and (5.51) give

$$\langle C_S^h \rangle = d_\ell \sum_{i=1}^{h} i + (m \bmod 2) \sum_{i=1}^{h} \frac{1}{2}\tau(-1)^{1+i}. \tag{5.52}$$

It can be conclude that by alternatively changing the additive value, the average time skew for networks with odd m decreases significantly. It should be noted that in any case and independently of m, the instant time skew of Eq. (5.43) is valid. Figure 5.23 depicts the $\langle C_S^h \rangle$ values calculated from Eqs. (5.48) and (5.51) assuming that $d_\ell = 0.1\,\tau$. Using the adjustment included in Eq. (5.51), the value of $\langle C_S^h \rangle$ for odd m is near to the values for even m and is much smaller than the unadjusted value obtained from Eq. (5.48).

5.2.3.3 Impact of Clock Drift and Update Interval

The validity of the calculated bounds for the clock time skew between nodes requires a process of updating and synchronizing the nodes periodically, since the accumulation of skew produces a clock offset that needs to be accounted for. For that, the appropriate updating moment has to be determined, as the time between updates depends on the expected clock drift.

Immediately after a successful synchronization, Eqs. (5.43) and (5.52) are valid. To estimate the effects of clock drift, suppose that the local system clocks have a normally distributed frequency with mean value $f_c = \frac{1}{\tau}$, and that nodes 1 and 2 initially have the same clock offset $T(0)$. The time difference due to clock drift after k clock cycles is

$$\Delta t_{2,1} = k\,(\tau_2 - \tau_1)\,, \tag{5.53}$$

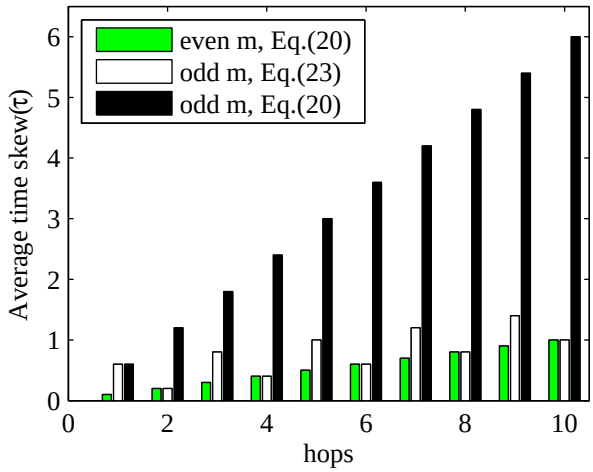


Figure 5.23: Average skew $\langle C_S^h \rangle$ as a function of hop count h.

where $\tau_1 = 1/f_1$ and $\tau_2 = 1/f_2$ are the system clock periods of nodes 1 and 2, respectively. Adjusting Eq. (5.52) to account for $\Delta t_{2,1}$ results in

$$\begin{aligned} \langle C_S^h \rangle = d_\ell \sum_{i=1}^{h} i + (m \bmod 2) \sum_{i=1}^{h} \tfrac{1}{2}\tau(-1)^{1+i} \\ + \langle \sum_{i=1}^{h} \Delta t_{i+1,i} \rangle \end{aligned} \tag{5.54}$$

Minimizing the effect of Δt requires reducing the value of k. The best case occurs when the *Sync* message arrives at the receiver just before the an active edge of *Clk-sync* is generated.

The necessary synchronization accuracy depends on the application. For a signal sampling rate of 1 kHz, a synchronization accuracy around 100 µs (10 % of the sampling rate) may be chosen. In many cases it will not be necessary to update the time in each period of $Clk - sync$. In general, the update interval can be determined by considering the required accuracy, Eq. (5.54) and the clock drift of the oscillators.

5.2.3.4 Probability of Synchronization

The entire process of synchronization is based on the successful reception of the timing messages. Consider the network of Fig. 5.19, where SN2 receives its synchronization messages from SN1. The probability of having both nodes synchronized is

$$p_{2,1} = P(t_2 = t_1) = p_{request} \times p_{sync}, \tag{5.55}$$

where $p_{2,1}$ is the probability of successful message exchange, which is the product of the probabilities for successful delivery of *Request* and *Sync* messages. The probability of message delivery depends on network traffic, topology and node or link failures.

For successful synchronization of a given node, all previous nodes on the path to the reference node must be synchronized.

Equation (5.55) can then be extended to a node located h hops away from the reference node:

$$p_{h,1} = \prod_{i=2}^{h} p_{i,i-1} \tag{5.56}$$

If the probabilities $p_{request} = p_{sync} = p$ are equal for all links, Eq. (5.56) simplifies to

$$p_{h,1} = p^{2(h-1)}. \tag{5.57}$$

Now consider the situation when PTP is used. This protocol uses 3 packets to complete the synchronization. So, the Eq. (5.58) for SN2 will be

$$p_{2,1} = p_{sync} \times p_{request} \times p_{response}, \tag{5.58}$$

where $p_{response}$ denotes the probability of a successful *Response* packet message transmission. Assuming that all messages have the same probability p of being successfully transmitted, Eq. (5.57) can be rewritten for PTP as

$$p_{h,1} = p^{3(h-1)}. \tag{5.59}$$

A comparison of Eqs. (5.57) and (5.59) shows that, for the application in wearable networks, PTP is more prone to failure in the synchronization process than the protocol proposed in this work, because PTP requires more messages to be transmitted.

5.2.3.5 General Protocol Operation

The aim of the proposed synchronization protocol is to ensure that a globally synchronized clock signal *Clk-sync* is available at every sensor node. The main purpose of this signal is to be used for synchronized sampling. In addition, each node maintains a counter that can be used to time stamp the acquired signals.

The simplest way to generate *Clk-sync* is to use an auto-reload counter: when the value of a down counter (called *TC* in the following description) reaches zero, an active transition of *Clk-sync* is generated and the counter is reloaded with a predefined value (called *TCPR*). The value of *TCPR* in all SNs must be the same.

Having all *Clk-sync* signals synchronized, is equivalent to say that all *TC* have to be synchronized. The counter *TC* is driven by the local system clock, which exhibits some clock skew relatively to the system clock of the other nodes. The procedure described in the previous subsection can be used to keep the average value of the skew within bounds. For this, each SN periodically updates its *TC* value with the *TC* value received from the timing message.

The current time stamp at each SN is maintained by an up counter *TS* (Time Stamping counter), which is driven by the synchronized clock signal *Clk-Sync*. The time resolution of *TS* depends on the frequency of *Clk-Sync*, which must be chosen according to the sensed phenomena: e.g., the

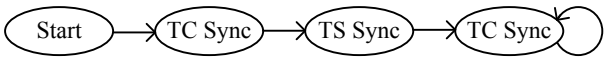


Figure 5.24: Sequence of MAC messages for clock synchronization generated by an SN.

EMG signals of [ZDD+12] have frequencies below 500 Hz, so a sampling rate of at least 1 kHz is necessary.

It should be noted that using *Clk-sync* for the *TS* counter only ensures that it counts synchronously. It is still necessary to synchronize the contents of *TS* at the start of the session with a timing message that includes the global current value of *TS*. Because each SN synchronizes its *Clk-sync* periodically, it is not necessary to do the same for *TS*: after a one-time *TS* synchronization, time stamping will stay synchronized while the system is running.

The process is summarized in Fig. 5.24. After starting, an SN first updates the *TC* counter for synchronization of the local *Clk-sync* signal; next, the *TS* counter is synchronized only once. Counter *TC* must then be synchronized periodically to compensate for the drift of the local system clock.

The protocol defines that synchronization starts when the slave node sends a timing *Request* message to the master node. Note that the same node can act as a slave (to synchronize its own clock signal) and as a master for several other nodes. The synchronization process executed in slave mode is shown in Fig. 5.25. The node starts by checking the availability of the connection to the master. If it is free, the slave node creates a *Request* in the MAC layer and sends it to the master. The request message specifies whether the request is for *TC* or *TS* synchronization. The slave node will then update its counters with the information received in the *Sync* message from the master node. The status of the received *Sync* message is saved in the *Status* register, so that it can be used in higher level processing by a microcontroller or microprocessor. For example, the value *Status* = *CRC* indicates that the *Sync* message was received with a CRC error. So, the timing information in the message is not valid and the receiver has to try again.

Master mode operation depends on the type of the *Request* message, as shown in Fig. 5.26. When a time stamp value is requested, the master simply replies with a message containing its own *TS* value. For a clock counter value request (*TC*), the master replies with the value of the difference *TC-Offset*, where *Offset* is the fixed value used to compensate the deterministic part of delay, which is fixed and equal for all sensor nodes. For the prototype described in Section 5.2.5, this value is set by software as part of the node's initialization sequence.

5.2.4 The Synchronization Circuit

This section describes a synchronization circuit that is based on the method just described. The circuit is part of the fully-digital sensor node communication system, which has been implemented both in an Actel low-power FPGA and in a CMOS ASIC.

Figure 5.27 shows the block diagram of the circuit including all necessary modules to send and receive timing information. The three modules in gray (*Signal Detector, TX Line SW and Line*

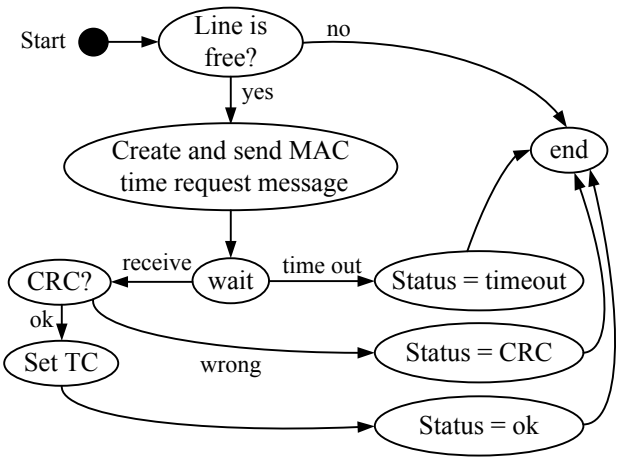


Figure 5.25: Sending the *Request* message and receiving *Sync* messages (slave node).

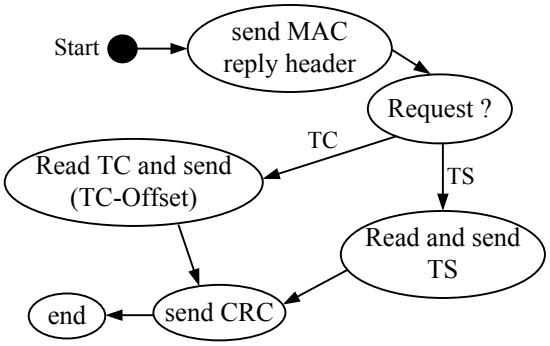


Figure 5.26: Replying to a *Request* message with a *Sync* message (in master node).

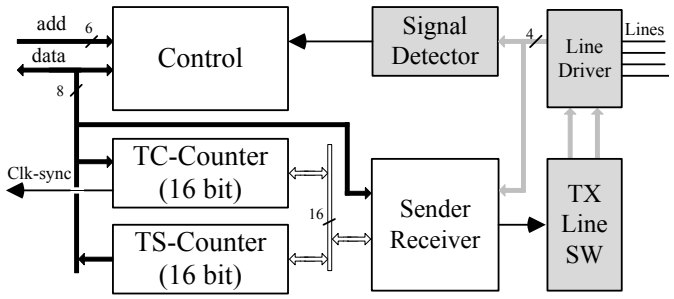


Figure 5.27: Block diagram of the circuit.

Driver) do not belong to the synchronization circuit, but they are involved in the communication process. An 8-bit internal system bus connects the module to the system core, which controls the time synchronization circuit, and reads or writes to the registers (e.g., the real-time time stamp). A 16-bit bus is used to connect the internal modules and to access the two counters *TC* and *TS*. The signal *Clk-sync* is available at one of the output pins. The synchronization circuit also manages a real-time time stamp that is available via the internal bus. The operation of each module is described next.

5.2.4.1 Control Module

The *Control* module manages all the activities related to time synchronization. It is connected to the system core by the internal 8-bit bus and also to the *Signal Detector*. All the register values can be read or written via this module. The main operation of *Control* is shown in Fig. 5.28.

Normally, this module is in sleep mode unless one of the following events occurs:

1. **Incoming timing message:** The SNs use several kinds of MAC messages, with timing synchronization being just one of them. The detection of a timing request message from an adjacent SN is carried out by the *Signal Detector* module. This module is responsible for determining the type of the MAC message by decapsulating and processing the MAC header, and forwarding the rest of incoming data to the destination module (which is the synchronization circuit for timing messages). For timing messages the MAC header also determines whether the request is for updating the *TC* or the *TS* counter. The *Control* module receives the timing message directly from the *Signal Detector* module and creates the appropriate reply message.

2. **Local command:** The request procedure for either *TC* or *TS* can be started by the node's microcontroller with a command sent via the SPI port to the *Control* module. The latter sets up the synchronization module for sending and receiving timing information, and starts the transmission. After sending the message, the *Control* module sets up the *Sender-Receiver* module to directly receive the MAC reply message.

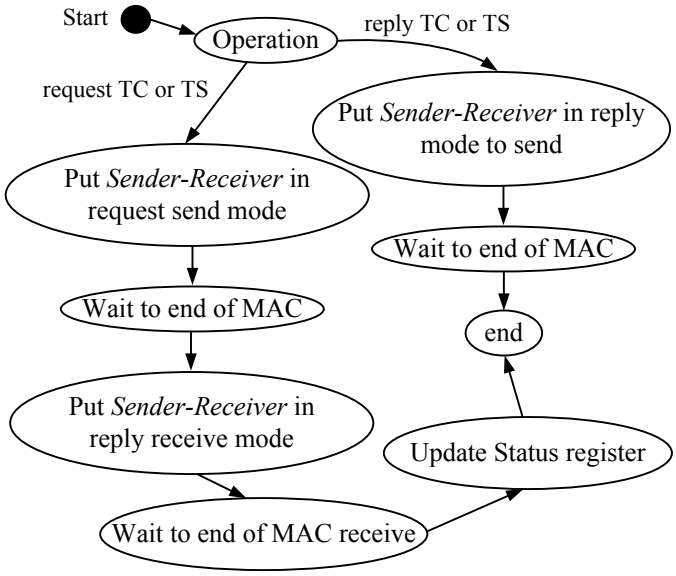


Figure 5.28: Operation of *Control* module.

5.2.4.2 TC-Counter and TS-Counter Modules

The *TC-Counter* module shown in Fig. 5.29.a generates the clock signal *Clk-sync* for time stamping and used as a synchronized clock by other parts of the sensors. Whenever the 16-bit down counter *TC* reaches zero, it is reloaded with the value of the 16-bit *TCPR* register. The periodic *Reload* signal is used to drive the clock generation circuit *Clock_gen*, which produces the reference clock signal *Clk-sync*. As mentioned before, this clock signal is used as input clock for the *TS-Counter* module, and is also available for use as clock reference in other parts of the sensor node. The output of this module is a pulse signal with duration of 1 or 32 system clock cycles. The wider pulses are necessary for some circuits. For example, when *Clk-sync* is used as an external interrupt of a microcontroller, a short pulse width may not be sufficient to generate a valid interrupt.

The wearable system in [ZDD+12] is designed to capture EMG signals using a 1 kHz sampling frequency. To generate the *Clk-sync* reference for the EMG signal sensing part, from a 20 MHz system clock , the *TC* counter must have at least a 16-bit , which is the length chosen for the current implementation. Therefore, the value of *TCPR* must be in range 1–65535 (33–65535 for the wide pulse mode). The frequency f_{sync} of *Clk-sync* is given by

$$f_{\text{sync}} = \frac{f_s}{1+\text{TCPR}}, \tag{5.60}$$

where f_s is the frequency of system clock. For $f_s = 20\,\text{MHz}$, the value of f_{sync} is in the range from 300 Hz to 10 MHz.

Figure 5.29(b) shows the *TS-Counter* module. This module includes *TS* and generates a time stamp that can be read by the system, or even written, via a 16-bit internal bus. As for *TC*, other SNs are able to obtain the value of *TS* for updates of their own *TS* counter. The time interval

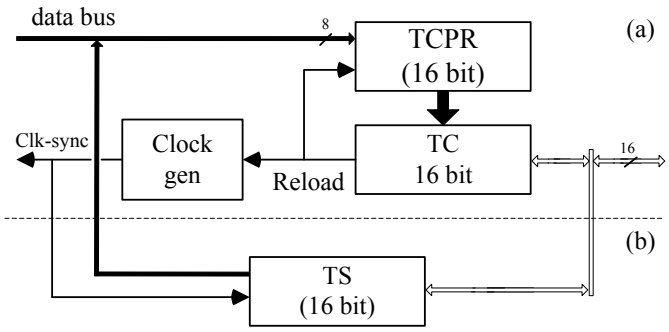


Figure 5.29: Counters: a) TC-Counter, b) TS-counter

between *TS* overflows is:

$$T_{\text{ov}} = \frac{65536}{f_{\text{sync}}}. \tag{5.61}$$

In fact, TS does not count the time ticks continuously, instead it is periodically reset every time interval $T_{\mathbf{ov}}$ (when overflow occurs). However, this will happen simultaneously in all nodes, which means that the information on relative event occurrence, between nodes, is always preserved. The base station can keep the absolute time count for all events.

5.2.4.3 Sender-Receiver Module

In our context, synchronizing two SNs is equivalent to ensuring that their *TC-Counters* have identical values at the same time. This ideal situation cannot be granted at all times, but a small bound on the difference between the counters may be enough for practical purposes. This is achieved by an exchange of special messages between the nodes. All processing of timing messages is performed by the *Sender-Receiver* module. It may operate in one of three different modes:

- **Request for timing information (Fig. 5.30):** start the synchronization process by sending a request message;
- **Reply to a Request (Fig. 5.31):** reply to a timing request message;
- **Reception of timing message (Fig. 5.32):** receive timing information in response to a request message.

5.2.4.4 Timing Request Message

In order to illustrate the operation of *Sender-Receiver* module, we will consider the situation where node SN3 in Fig. 5.19 needs to synchronize its *TC* value with its *near-node* SN2.

The process is started by the upper layers of the control software of node SN3 by sending a command (via the internal bus) instructing the *Control* module to activate the request mode. The *Control* modules instructs the *Sender-Receiver* module to start communication with SN2. The configuration of the module in this mode is shown in Fig. 5.30.

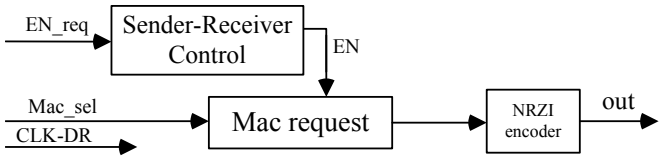


Figure 5.30: Configuration of *Sender Receiver* to send a request.

The *Control* module generates both of the *EN_req* and *Mac_sel* signals: the first enables the request mode and the second specifies that the request concerns the value of *TC*. Based on the request type (which could be *TC* or *TS*), the *Mac_ Request* module generates the appropriate MAC message, which must then be encoded for transmission. The line encoding scheme used in our implementation is NRZI. Therefore, the output of *mac request* is encoded before driving the line. The module *TX SW* (Fig. 5.27) forwards the MAC message to the line connected to SN2. At the end of the process, the *Control* module changes the configuration of the *Sender-Receiver* module to reception mode and waits for a reply from SN2.

5.2.4.5 Timing Reply Message

When the *Signal Detector* module of SN2 detects the timing message request, it forwards the incoming data to the synchronization circuit. The *Control* module enables the reply mode by asserting the *EN_rep* (Enable-reply) signal. In this mode, node SN2 must send the value of its *TC* counter to SN3, after performing an adjustment to account for the offset between the nodes. The configuration of the *Sender-Receiver* module is the one shown in Fig. 5.31. In that figure, *B-TCTS* is a 16-bit register for buffering the sampled value of *TC* in node SN2 at the start of reply processing. The 16-bit register *Offset* is used to memorize the offset value, which will be used to compensate for the skew introduced by the communication delay. The value of *Offset* can be managed by the upper software layers through the *Control* module. Module *Sub* is a serial subtracter that is used to subtract *Offset* from the *TC* value buffered in *B-TCTS*. According to Eq. (5.46), the *Offset* value must be added to the *TC* value that is sent back to SN3. However, *TC* is a down-counter, so the *Offset* value must be subtracted instead. It should be noted that a 1-bit serial subtracter can be used to perform the subtraction of two 16-bit values, since communication is serial and the subtraction can be performed during reception.

To protect and check the validity of the data at the receiver node, the *CRC5* module generates a 5-bit cyclic redundancy check, which is concatenated to the outgoing data. A multiplexor controlled by *Send-Receive Control* selects, in order, the *Mac reply*, *Sub* and *CRC5* modules to build the full reply message at the output, which goes through the NRZI encoder before being sent to node SN3.

5.2.4.6 Reception of Timing Messages

As mentioned previously, node SN3 changes the configuration of the *Sender-Receiver* module after sending the timing request message. The configuration for reception is shown in Fig. 5.32.

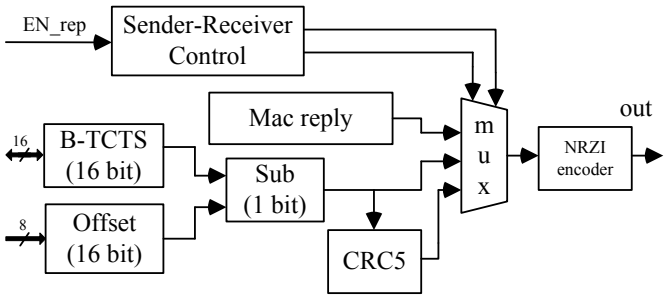


Figure 5.31: Configuration of *Sender-Receiver* for reply to a timing information request.

In this case, register *B-TCTS* acts as a Serial-In-Parallel-Out buffer to receive the *TC* value. After the *CRC5* module confirms the validity of the received data, the value buffered in *B-TCTS* is loaded to the *TC*.

The processor for updating *TS* is the same as the one used for *TC*, with the only difference that, in the reply step, the value of *TS* will be sent without any adjustment. As mentioned in Section 5.2.3.5, *TS* counters inherit synchronization from *Clk-sync* signals; exchanging *TS* values is necessary only for time stamp alignment.

5.2.4.7 Implementation characteristics

The time synchronization module in the fabricated IC is highlighted in Fig. 6.19. The box labeled *TS* indicates the approximate area occupied by the synchronization circuit. It uses 971 cells (765 gates and 206 Flip flops) and occupies 0.1388 mm^2 (19 % of IC), a number which is significantly smaller than the 15115 cells required by the hardware implementation of the more complex IEEE-1588 protocol reported in [PHC+11].

5.2.5 Experimental Results

This section presents experimental results obtained with the IC version of the circuit. The system clock frequency of each node is 20 MHz, which results in a data rate of 10 Mbps. With this system clock frequency, the synchronized clock can be set in the range 305 Hz–10 MHz. The network of SNs used for the measurements is composed by the nodes BS, SN2, SN3, SN4, SN5 and SN6 of Fig. 5.19.

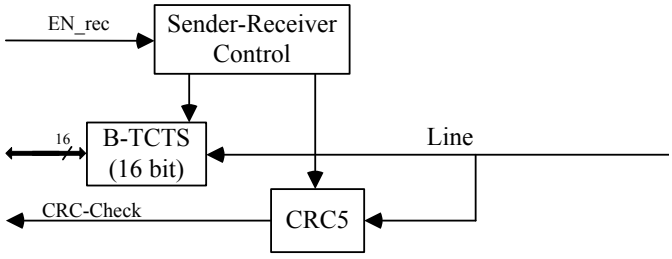


Figure 5.32: Configuration of *Sender Receiver* to receive timing message.

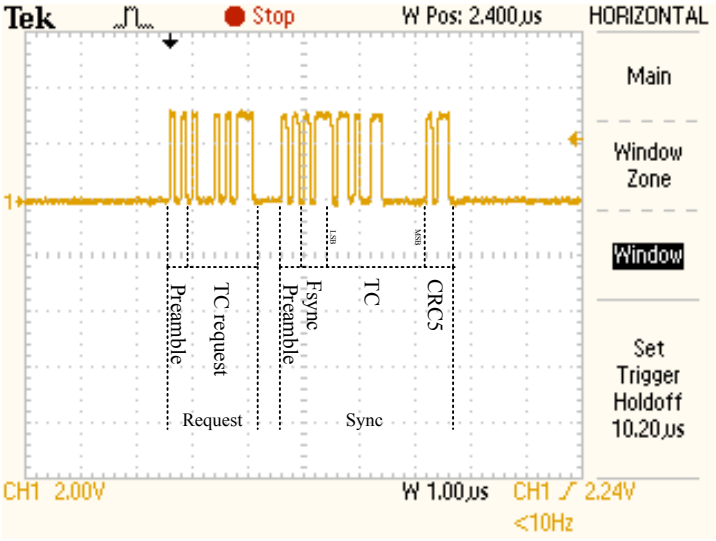


Figure 5.33: Physical layer signals during timing message exchange between SN2 and BS.

5.2.5.1 Physical Layer Signaling

Signals generated in the physical layer during message exchange (request and reply) for synchronization of the *TC* are shown in Fig. 5.33. As mentioned before, the line between the nodes is bidirectional, so the reply message appears on the same line immediately after the request message. The Least Significant bit (LSB) of the adjusted *TC* value is sent first; the message trailer consists of the CRC check bits.

For the evaluated setup, the entire process, from sending the request message to the end of the reply message, takes 5.2 µs. To keep network nodes synchronized, it is necessary to exchange timing messages periodically, which requires the utilization of system resources, such as channel time, and should be take into account. For an interval between timing messages of 1 ms only 0.52 % of the channel time will be used for synchronization, a very small percentage of the total traffic on the network.

Figure 5.34 presents part of a logic simulation showing the *TS* synchronization of SN3 and SN4 for a scenario in which *Clk-sync* is configured to operate at 81 kHz. The simulation shows precisely when the changes of the *TS* counter occur. This internal information cannot be obtained by observing the external IC signals. The final *TS* value can be read via the SPI port, but its instantaneous value is not externally accessible. In the figure, *L* indicates the state of the communication line between the two nodes. For each of the nodes, the simulation shows the *Clk-sync* signal and the contents of the *TS* counter.

At the beginning of the simulation, both *Clk-sync* (generated by *TC*) and *TS* are out of synchronization. The synchronization of *TC* starts at $t = 205.7\,\mu s$, when node SN4 sends a *Request* message to SN3. The round trip message takes 5.2 µs to complete (ending at $t = 210.7\,\mu s$). The receiver takes 50 ns (one system clock cycle) to check the validity of the message and loads the receiver's *TC* counter at $t = 210.7\,\mu s$, resulting in a synchronized *Clk-sync* at $t = 220\,\mu s$.

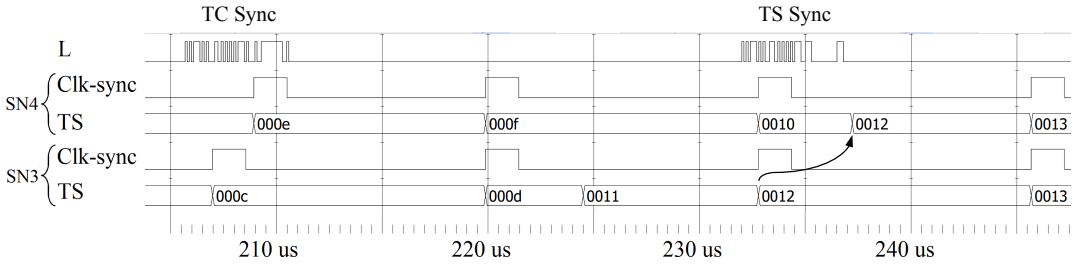


Figure 5.34: Simulation showing synchronization of *Clk-sync* signal and *TS* counter.

To synchronize the time stamp *TS*, node SN4 sends a request at $t = 232$ ns. Node SN4 replaces its *TS* value with the value received from SN3 (which is 0x0012) at time $t = 237.2\,\mu$s. So, after 237.2 µs both nodes are synchronized. A further noteworthy aspect of the simulation is that during the previous process node SN3 has synchronized its *TS* value with node SN2 at $t = 224.5\,\mu$s (new value 0x0011).

5.2.5.2 One-Hop and Multi-Hop Clock Skew

The one-hop clock skew between SN2 and BS for various values of the parameter *Offset* can be seen in Fig. 5.35. The oscilloscope signals shown in the figure are generated by using the "infinite time persist" display mode. The system clock of the BS is used as reference in all skew measurements. The offset value for compensating the constant delay value was measured to be $s =$ 0x0030. The observed clock skew variation range is 50 ns, that is one period of the system clock, as described by Eqs. (5.40) and (5.41). The measured average one-hop clock skew is 54 ns, i.e., the sum of signal propagation delay and $S_r(t)$. The measurements confirm that the synchronization circuit is able to keep the clock skew in the bounds defined by theoretical analysis.

By setting the *Offset* parameter to $s' = s - 1 =$ 0x002F, the *Clk-sync* shifts one clock cycle to the left, causes to decrease the skew. In the setup used for the measurements, the average clock skew is 4.6 ns, which is much smaller than the the total clock skew variation of 54.6 ns. This value is the minimum skew achievable by the proposed method and shows the feasibility of minimizing the average clock skew by selecting appropriate values for the offset. The effect of using the offset value 0x002E is also shown in the Fig. 5.35, with one more clock period shift as expected. So, by changing the offset value it is possible to shift the *Clock-sync* in both directions by an arbitrary value. Regardless of the offset value and its effect, the peak-to-peak clock jitter is 50 ns.

The clock skew increases with the number of hops. Figure 5.36 depicts the clock signal at different nodes (the offset value, 0x002F, is the same for all the nodes). As can be seen, and as expected, the average clock skew increases by almost 4.6 ns as the number of hops increases. In agreement with Eq. (5.41), the clock skew variation range also increases: 50 ns at SN2, 100 ns at SN3 and SN5, and 150 ns at nodes SN4 and SN6. The comparison between the measured and calculated values is shown in Fig. 5.37, confirming the correct operation of the circuit.

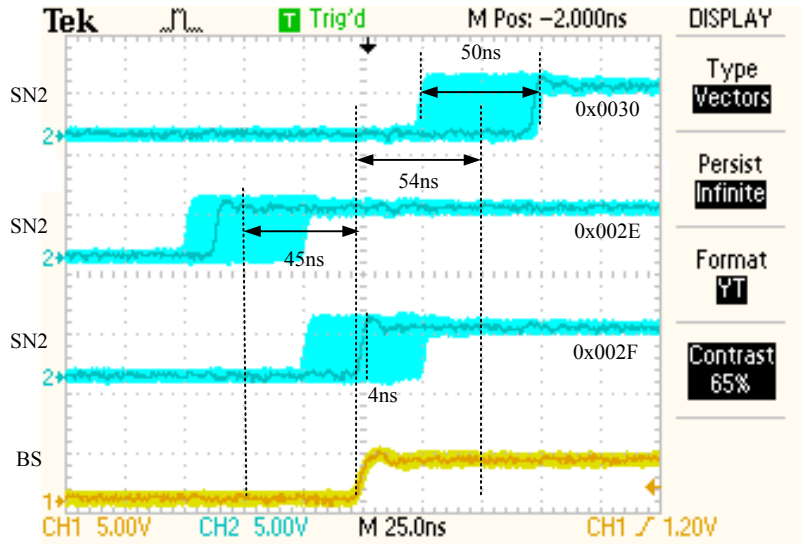


Figure 5.35: Measured one-hop clock skew for Offset =0x002E, 0x002F and 0x0030.

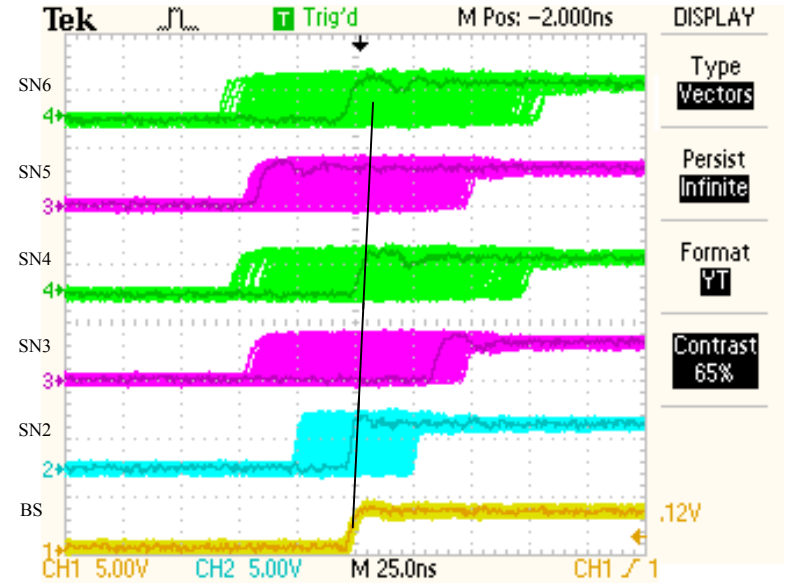


Figure 5.36: Measured multi-hop clock skew with *Offset* = 0x002F.

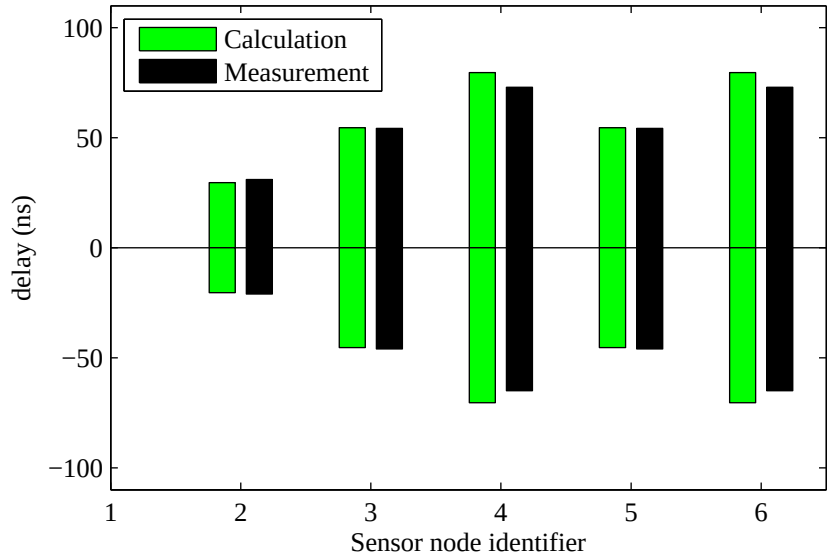


Figure 5.37: Comparison between the measured and calculated values for multi-hop clock skew with *Offset* = 0x002F.

5.2.5.3 Power Consumption

Figure 5.38 depicts the measured current consumption of the circuit as a function of the *Clk-sync* frequency from 305 Hz to 5 MHz. The current increase linearly from 0.18 mA to 0.64 mA as the frequency increases. With *Clk-sync* = 1 kHz the total current is almost 0.18 mA. Most of the current is used to drive the *Clk-sync* pin that is available as an output of the ASIC.

5.2.5.4 Effects of Timing Message Interval and Failure on Synchronization

In many applications, message failure is inevitable and it may happen for several reasons such as collisions and high traffic load. To evaluate the effect of message failure on synchronization, the clock skew between BS and SN2 has been measured as a function of the number of consecutive

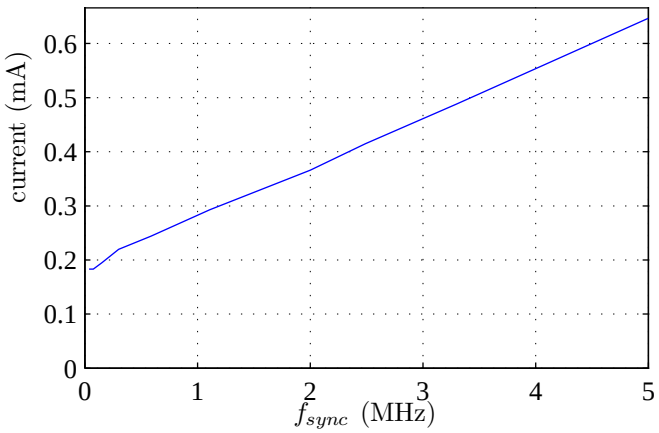


Figure 5.38: The circuit current consumption for f_{sync} up to 5 MHz

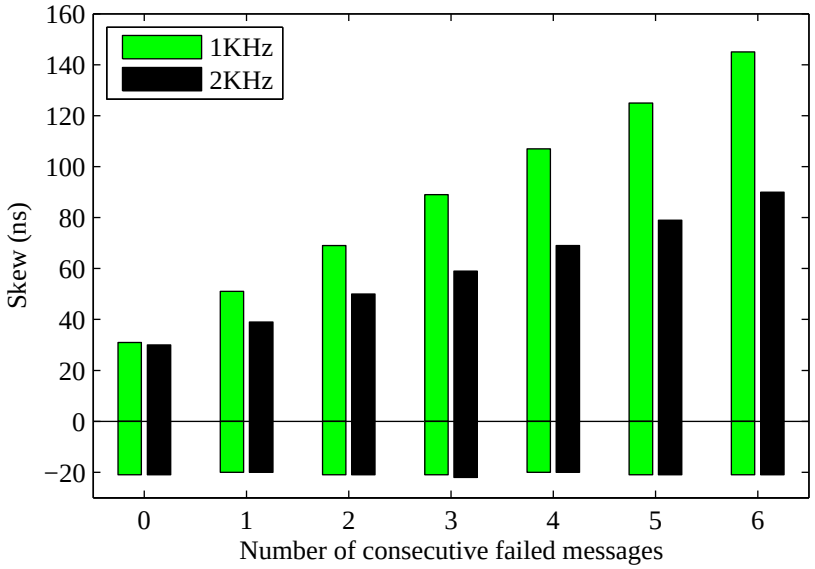


Figure 5.39: One-hop skew after consecutive timing message failures.

missing messages. The results are shown in Fig. 5.39 for two different *Clk-sync* frequencies. In both cases, the timing update messages are sent just before the rising edge of *Clk-sync*. Therefore, the interval between updates used for the 2 kHz clock signal is half the interval used for the 1 kHz case. Therefore, the clock skew for the latter case is larger than the skew for the 2 kHz case.

According to Eq. (5.53), the linear growth of clock skew depends on the clock frequency difference between the nodes. In the absence of failures, the synchronization accuracy is in the range [−21 ns; 29 ns], as expected. The observed increase of the clock skew in the presence of failures means that, in this case, $\Delta t > 0 \Rightarrow \tau_S > \tau_M$. This means that the upper (positive) bound of the skew grows with the number of missing messages, while the lower (negative) bound stays the same. For situations where $\Delta t < 0$, it is the lower bound that becomes increasingly more negative. Regardless of the clock skew, a successful synchronization message puts the skew of the receiver back in the range [−21 ns; 29 ns].

If a line between two nodes disconnects or breaks, we expect the skew of the disconnected segment to grow without limit, but the synchronization between nodes in each segment to be maintained. To confirm this behavior, nodes SN2 and SN3 were disconnected at $t = 37$ s in order to obtain the results for average clock skew shown in Fig. 5.40. Until the disconnection occurred all nodes stayed synchronized. At $t = 37$ s SN3 starts running free and its skew starts to increase as expected. The nodes on the disconnected segment get their time reference (directly or indirectly) from SN3 and stay synchronized with it. The clock skew after the disconnection depends on the clock difference between BS and SN3. In the present case, the positive slope of the skew curve is due to the fact that the clock of SN3 has a somewhat higher clock frequency that the clock of BS. In the clock of the disconnected node has a smaller frequency, the slope of the skew curve will be negative.

The periodic exchange of timing messages generates additional traffic. Measurements show

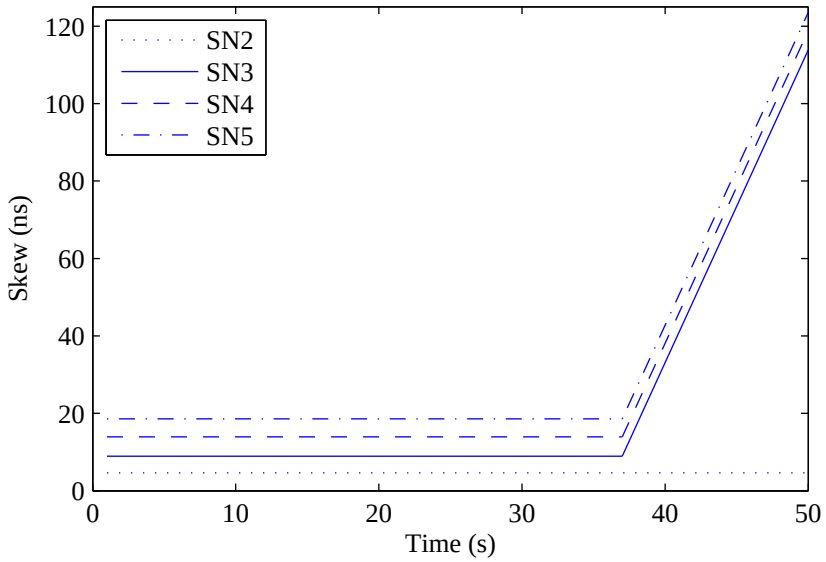


Figure 5.40: Average clock skew after disconnection of the line between SN2 and SN3.

that with a 1 ms interval between synchronization messages, 0.52 % of the channel bandwidth is used for synchronization, which is almost negligible. This allows the system to achieve high precision with an average clock skew below 5 ns. Some BAN applications may not need such a high precision. If precisions in the range of a few microseconds is enough, the time interval between messages can be larger (in the range of a second), consuming even less energy and channel bandwidth.

Figure 5.41 shows the measured skew between BS and SN as a function of time for two update intervals: 1.5 s and 5 s. The skew depends on the clock frequency difference between the nodes, which has been measured to be 3.7 ppm. Since the clock frequency of each node stays constant, the accumulation of the small difference grows linearly, reaching 18.5 µs (for a 5 s interval) and 5.58 µs (for a 1.5 s interval). In both cases, the absence of synchronization message for a long time results in significantly higher skew. Nevertheless, a message synchronizes *Clk-sync* immediately.

The measured value of 4.6 ns is due to the signal propagation from sender output node to the input port of receiver node, an unavoidable inherent characteristic of the communication path (I/O pads and connecting yarn). In resource-limited systems, such as sensor networks, achieving a smaller clock skew is difficult in practice, even if more sophisticated two-way methods are used, as it would require more processing and a more complex implementation, Therefore, the low-overhead approach described here makes it practical for many wearable systems to achieve a very-low skew in range of sub-microsecond (4.6 ns for the wired, yarn-based application used for evaluation) with a one-way method and without any further processing.

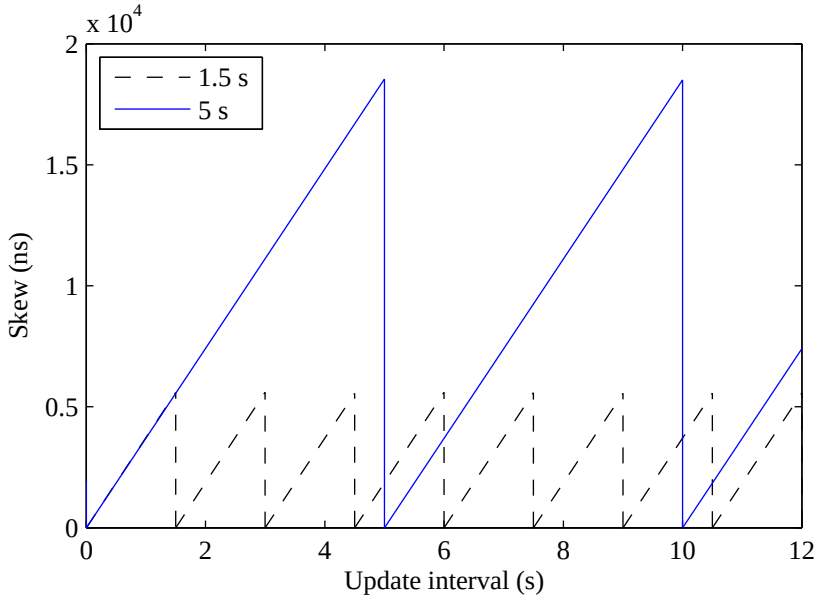


Figure 5.41: Clock skew for update interval of 1.5 s and 5 s, *Clk-sync* = 1 kHz.

5.2.6 Conclusion

The circuit described in this section was designed for establishing time synchronization between sensor nodes of the wearable part of the system. The synchronization is based on one-way master-to-slave message exchange implemented in the MAC layer, in order to avoid the non-deterministic delays caused by data processing and buffering in the higher levels of the protocol stack. By directly sending and processing the timing information without buffering, the proposed approach leads to an average clock skew of a few nanoseconds. The circuit generates two synchronized values: a programmable clock signal and a real-time counter for time stamping purposes.

Experimental evaluation with IC implementation obtained an average one-hop clock skew of 4.6 ns that is the time required for signal propagation from sender output to the receiver input. Based on theoretical calculations, in a multi-hop network, the global average time skew grows linearly with hop count; this is supported by the experimental results. The low skew values provided by this approach satisfy the requirements of many BAN applications. Even for networks whose nodes are 10 hops away from the time reference node, the average global skew will typically be under 50 ns. A value of 10 hops exceeds the largest inter-node distance of many, if not all, existing wearable systems. The proposed circuit achieves the maximum synchronization performance that could be achieved by PTP, but with fewer timing messages and calculations, less complexity and better energy efficiency.

Chapter 6

Architecture and Implementation of a Communications ASIC

Results from the first prototype described in Chapter 4 gave a few indications on ways to improve the system performance. In order to address those issues, the present chapter introduces the second prototype based on hybrid circuit and packet switching implemented in an ASIC. The circuit links are established at the MAC layer level over the minimum cost paths provided by the SRMCF protocol. So, circuit switching is based on cross-layer information exchange. The new architecture provides less power consumption, reduces buffering requirements in intermediate nodes, and improves end-to-end delay and throughput. The ASIC also contains a high precision time synchronization module implemented in the MAC layer. To reduce the packet serving time, the routing of data packets sent from SNs to the BS is implemented inside the IC. This chapter also presents an analysis of circuit switching in the context of wired BANs, a TDM protocol scheduling transmissions, the functionality of the integrated communication circuit and its modules, and the experimental results obtained from sensor nodes equipped with the new ASIC.

6.1 Bufferless Communication and Circuit Switching

In networking, a switching algorithm, such as packet switching, specifies the end-to-end packet delivery method. Collisions happen in shared environments when two or more nodes try to use a common channel or any other resources. In packet switching each node stores and forwards the packets whenever the links are free. The results from the previous chapter show that employing packet-switching in a wearable system may adversely affect power consumption, throughput and end-to-end delay.

One of the methods that can overcome the aforementioned drawbacks is to use circuit switching at MAC layer, because it is a bufferless method. In bufferless networks, such as optical networks or some Network-on-Chip (NoC) architectures, a packet arrives at the destination node over intermediate nodes without any buffering [BMIM07, ZGY14, MM09, GAM+12]. This approach usually reduces the total network power consumption and delay under low network workloads,

but routing performance for randomly generated packets is significantly worse than for traditional buffered routing (store-and-forward method) for high network workloads. In fact, a bufferless algorithm without communication scheduling at the nodes may be significantly ineffective. The following sections introduce a bufferless method based on circuit switching for many-to-one packet delivery, which is the situation that applies to node-to-BS data communication; other communications, such as node-to-node, are still based on packet switching. So, communication is a hybrid of both circuit and packet switching. An algorithm for time scheduling at the nodes is also introduced. The results presented in Section 6.4 shows that system performance by scheduling of the nodes is better than packet switching, close to ideal network performance.

6.1.1 A Time-Division Multiplexing MAC Protocol Based on Circuit Switching

As mentioned earlier, in a network based on circuit switching, randomly generating and sending packets leads to a significant degradation of the throughput because of the high number of request failures in shared nodes and links. To avoid this and for a higher likelihood of path construction and packet delivery, a time scheduling method is essential, specially in very loaded networks. In this case, the operation time of the system is divided in slots with the appropriate period to transmit messages and a packet in each time slot. Such a scheduling method allows each node to use the shared environment for sending packets without collision. This method will be effective when the nodes obey the time scheduling by using a global synchronization method. Any time skew due to synchronization error may strongly affect the network performance.

Let $M = \{n_0, n_1, n_2, ..., n_m\}$ denote a set of nodes connected in a mesh topology and let n_0 represent the BS node. A routing protocol like SRMCF determines a spanning tree with the BS at the center. When employing TDM, all nodes send packets according to a time schedule. The design of a scheduling algorithm needs to take two aspects into consideration:

1. *Sharing the nodes and links:* All the nodes and links are shared. To share the links and nodes, each sender node first sends a request message (RTS) to its *near-node* for allocating a free path; then that *near-node* sends another RTS to the its *near-node*, and so on until the BS is reached. To establish a path, then several RTS message are needed. Circuit construction is explained in Section 6.1.2.

2. *Variable length of total RTS:* The number of RTS messages required to define a circuit path depends on the hop distance from the sender node to BS. For example, a node connected directly to the BS needs one RTS, whereas a node at a distance of four hops needs four RTS messages.

Dependency of the total number of RTS on the hop count affects the time slot duration. Assume that all time slots have equal duration. In a simple method, the duration of time slots has to be defined as to carry all the RTS messages together with CTS and a data packet. So each time slot will take

$$ts = h_{max} \times T_r + T_s + T_P + 3\tau + T_g, \tag{6.1}$$

where ts denote to the time slot duration, h_{max} maximum hop count, τ propagation time including the signal propagation and detection time, T_r RTS time, T_c CTS time, T_P total packet time, and T_g a gap between the slots (for protecting communication from overlapping the time slots). The value of h_{max} must be selected so as to ensure that any fixed-length time slots is able to fit a communication between any node and the BS. By using this method and allocating a time slot to each node, any arbitrary node communicates with BS without any overlap with others. Although allocation of time slots can be easily done, it should be noted that the term $h_{max} \times T_r$ imposes an overhead on message exchange and consumes network time. Consider a node one hop away that wants to send a packet to the BS in its allocated time slot. The required time is $T_r + T_s + P + 3\tau$. So, $(h_{max} - 1) \times T_r$ time of time slot will not be used, which has a detrimental impact on throughput, specially in networks with a big h_{max}.

$$S = \frac{T_P}{h_{max} \times T_r + T_s + T_P + 3\tau + T_g} \tag{6.2}$$

where S denotes the throughput. So, h_{max} must be as small as possible, which imposes the constraint of making the network as compact as possible. The hop count value in compact networks is explained and evaluated in Section 4.1.3. For a network comprising m four-port nodes, that analysis implies that

$$h_{max} \geq \left\lfloor 1 + \log_3 \frac{m+1}{2} \right\rfloor \tag{6.3}$$

To overcome the effect of variable numbers of RTS on the network performance, two methods can be applied:

1. *Variable duration slot:* The allocated time for each node depends on the hop distance. So, the time slot ts_i assigned to node i at hop distance h_i will be

$$ts_i = h_i \times T_r + T_s + T_P + 3\tau + T_g \tag{6.4}$$

   By using variable slot durations, the throughput of the system will increase. However, it should be noted that variable time-slot lengths may become difficult manage, specially in a large network.

2. *Scheduling of the nodes*: Regardless of using fixed or variable time-slots, their allocation to each node is arbitrary, since there is no overlapping of time slots. Although BS is able to receive one packet in any given time, it does not matter from which sensor the packet is coming from. In any case the BS receives only one RTS message. However, it is possible to exploit the fact that some other nodes are able to start constructing the path to the BS even while the BS is receiving a packet from the currently scheduled SN. Exploiting this possibility reduces the effect of RTS overhead and increases throughput.

The second option is that proposed in the present work: an hybrid approach of circuit and packet switching with fixed-duration slots, for simpler management. The following sections explain first the circuit path creation method and then the adopted scheduling method.

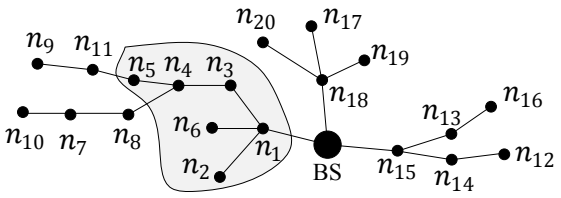


Figure 6.1: A random generated network with 3 subset of the nodes connected to the BS.

6.1.2 Circuit Path Construction

Before starting a data packet transmission, a path has to be formed between sender and destination nodes. All the links and nodes involved in a path will be reserved for the duration of the transmission and released at the end. All paths are predetermined minimum cost forwarding paths generated by the SRMCF protocol in the network setup phase, as explained in Section 3.1.2. Regardless of the selected switching method, the communication in the direction from BS to the SNs is always establish by packet switching.

Consider the situation in Fig. 6.1 when node n_4 wants to send a packet to the BS without buffering in the intermediate nodes. Figure 6.2 depicts the network operations when employing circuit switching between n_4 and the BS for delivering of packet *PKT*. First of all, n_4 has to inform n_3 about establishing circuit switching. For that purpose, n_4 sends a request message *RTSnd* to its *near-node* n_3. The *RTSnd* is the RTS message used to establish circuit switching; it also carries the hop count (c in the figure). The hop count is initially one and is incremented by one at each node. It is used to control the number of intermediate SNs involved in the circuit. Circuit switching reduces buffering and end-to-end delay, but if the number of the intermediate nodes increases to much it may degrade the network performance because many resources (links and nodes) are tied up in the circuit. So, the hop count information may be used to control the length of the path (as described in Section 6.2.5).

Now, if n_3 is ready, it sends back a waiting message to n_4 (shown as w in the figure) to indicate that it is ready and is going to continue with the process of establishing the circuit by sending another *RTSnd* to its own *near-node*, which is n_1. Then n_3 increases the c field, sends an *RTSnd* message to n_1 and sets up its own switch so as to connect directly the lines to nodes n_4 and n_1 (step 2). In this way, any signals from n_1 appear in the line connected to n_4 after a very small delay (only due to the signal propagation on the internal path between the two ports).

Then n_1 sends a *RTSnd* request to the BS, and a wait message to nodes n_3 and n_4 (through the partially established circuit) (step 3). It also sets up its own switch so as to link the ports connected to n_3 and the BS (in the same way as is done at node n_3).

After this step, n_4 is connected directly to the BS via intermediate relays (n_1 and n_3) and receives almost immediately any signals that BS generates. If BS is free to accept, it replies with a CTS message (step 4). Nodes n_1, n_3 and n_4 receive the *CTS* message almost simultaneously. Then nodes n_1 and n_3 change the direction of their switches to allow n_4 to start sending the packet to the BS (step 5). At the end of packet transmission, all nodes release the lines, but if *ACK* message is

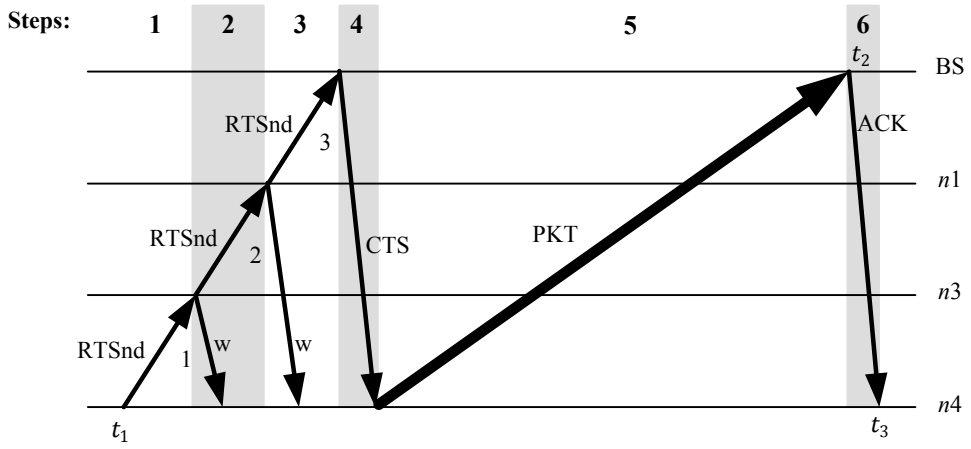


Figure 6.2: Routing a packet from n_4 to BS by using circuit switching.

necessary, then nodes n_1 and n_3 first change again the direction of their switches to hand over the *ACK* message from BS to n_4 (step 6). After that, all nodes end the communication and release the lines.

As is observable, the packet is not buffered at the intermediate nodes and the end-to-end delay in this case is $t_3 - t_1$, which is much smaller than the time required for packet switching.

6.1.3 Scheduling Node Transmissions

In this section, a scheduling method to share and optimally use of the transmission capacity is presented. A round-robin scheduling approach is adopted, where each node has a chance to construct a circuit path in the dedicated time slot according to a pre-assigned repeating order. The assignment is done by the BS. All time slots have the same duration and the effect of the *RTSnd* message overhead on the network performance is optimized. First of all, the routing protocol SRMCF finds a spanning tree with BS at the center. Assume that the BS is connected to the subset of nodes

$$m_i \in M, \;\; i = 1,2,3,4$$

Figure 6.1 depicts an example of such a network with 3 subsets:

$$m_1 = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11}\}$$

$$m_2 = \{n_{12}, n_{13}, n_{14}, n_{15}, n_{16}\}$$

$$m_3 = \{n_{17}, n_{18}, n_{19}, n_{20}\}$$

Assume that n_1 in Fig. 6.1 is sending a packet in the time slot TS_1 as shown in Fig. 6.3. At the same time, n_{12} from subset m_2 has a packet ready to be sent. Normally, n_{12} would wait for the next time slot (TS_2). However, since n_1 and n_{12} are in separated branches, then n_{12} can start to send its

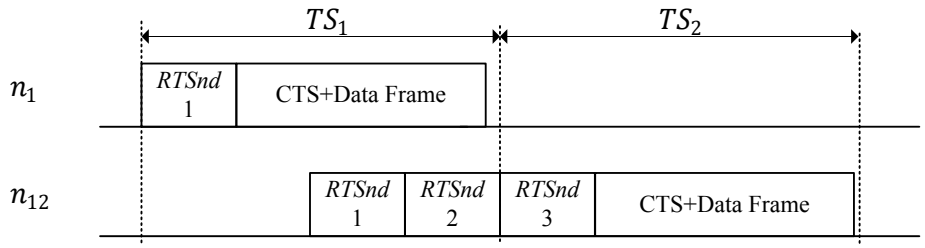


Figure 6.3: Time slot assignment for nodes n_1 and n_{12}.

RTSnd message before TS_2 starts, in such a way that the 3rd *RTSnd* appears at the beginning of TS_2. BS will still receive the packet without any collisions with the one generated by n_1.

In essence, any member of the set of nodes

$$\{1 \times RTSnd, \quad \forall(n_r, n_s) : n_r \in m_i, n_s \in m_j \mid i \neq j \} \tag{6.5}$$

is able to use the time slots in the same way as explained for n_1 and n_{12}.

To begin assigning nodes to time slots, one of n_1, n_{15} or n_{18} (the nodes directly connected node to BS) has to be selected, because they need only one *RTSnd* message. It is clear that selecting any one of the three nodes does not affect the final performance. Nevertheless, to account for the situation where a set is simply a line of nodes (discussed later), it is preferable not to select first n_1, because it belongs to the largest set. For this example, consider that n_{15} is selected. To increase the usable time of time slots, the next node must be selected from sets m_1 or m_3, preferably the unscheduled node that is located farthest way from the BS. The time slot allocation can be done in many ways. For the network example of Fig. 6.1 all nodes in

$$\{n_{15}, n_{10}, n_{17}, n_{12}, n_9, n_{19}, n_{16}, n_{11}, n_{20}, n_{14}, n_7, n_{18}, n_{13}, n_8\} \tag{6.6}$$

m_1 : ↑ ↑ ↑ ↑ ↑

m_2 : ↑ ↑ ↑ ↑ ↑

m_3 : ↑ ↑ ↑ ↑

need to have only one *RTSnd* in their allocated time slots.

The remaining unscheduled nodes are from m_1, since it as the largest set (highlighted set in Fig. 6.1):

$$m_{1r} = \{n_1, n_2, n_3, n_4, n_5, n_6\}$$

For this example, the number of the remaining unscheduled nodes is

$$m_1 - max(m_2, m_3) = 11 - 5 = 6.$$

Because the far nodes are selected first, m_{1r} is a subset of m_1, including the nodes nearer to the BS that share the path through n_1.

The aforementioned method can be applied to those nodes that share intermediate nodes in the

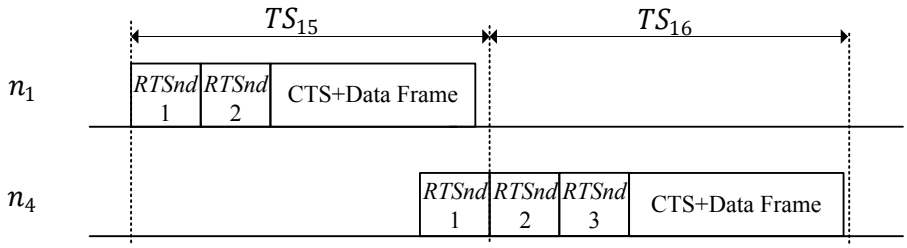


Figure 6.4: Constructing paths for nodes n_2 and n_4.

path to the BS, e.g., both paths from n_2 and n_4 have n_1 in the way to the BS (Fig. 6.1). In this case, if n_4 sends a packet after n_2, then it has to be sure that their common node (n_1) is free. So, the time slot of n_4 contains two *RTSnd*, one from n_3 to n_1 and another from n_1 to BS. In fact, the situation is similar to that of the nodes in the set defined by Eq. (6.6), except for the common node n_1, it causes an extra *RTSnd* message in each time slot as shown in Fig. 6.4. Since all time slots have a fixed size, then packet n_4 in time slot TS_{16} has to be one T_r (for extra *RTSnd* time) smaller than the packets for nodes in the set described by Eq. (6.6).

It can be concluded that node n_1 plays the same role for the nodes in set m_{1r} as the BS plays for nodes in the set Eq. (6.6). Hence, the same method can be used, but this time taking n_1 as the BS. In this case, the result of the assignment procedure will be the set of nodes $\{n_2, n_5, n_6\}$ that need two *RTSnd* in their time slot. So, for the nodes in m_{1r} (except n_1)

$$\{2 \times RTSnd, \quad \forall(n_r, n_s); n_r \in m_{1ri}, n_s \in m_{1rj} \,|\{m_{1ri} \in m_{1r},\ m_{1rj} \in m_{1r},\ i \neq j\}\}. \tag{6.7}$$

The remaining nodes n_1, n_3, n_4 are on a single line. Because of the absence of branches, the situation in such a set of nodes is different. In general, let $L = \{b_1, b_2, \ldots, b_l\}$ denote nodes in one line with b_1 one hop away and b_l at a distance of l hops. The first sender is the one nearest to BS, which is b_1. At the same time all other nodes have opportunity to prepare the path for transmission in the next time slot. To use the maximum time gap, the best selection will be to scheduled the farthest node (b_l) first. Because all nodes are involved in the circuit built by b_l, other nodes have to wait until the end of the time slot. At the beginning of the next time slot, all remaining nodes are in the position to send a packet, but the best selection will be b_2 as the unscheduled node nearest to the BS. This process is repeated for all nodes. So, the final sequence will be

$$b_1, b_l, b_2, b_{l-1}, ..., b_{\lfloor \frac{l}{2} \rfloor + 2}, b_{\lfloor \frac{l}{2} \rfloor + 1} \tag{6.8}$$

where $b_{\lfloor \frac{l}{2} \rfloor + 2}$ and $b_{\lfloor \frac{l}{2} \rfloor + 1}$ denote the nodes in the middle of the line. The number of *RTSnd* messages for the aforementioned sequence is

$$1, 2, 2, 3, 3, ..., \lfloor \frac{l}{2} \rfloor + 1$$

In summary, the proposed time slot assignment algorithm performed by BS is the following:

1. **Create all the paths:** In the first step, the network goes through the setup phase of the SRMCF to make all minimum cost paths and create spanning trees of nodes with the BS at the vertex (the setup phase is explained in Chapter 3).

2. **Determine connection matrix and hop counts:** Each node receives packets from several nodes but sends them only to one node (to its *near-node*). On the other hand, the nodes and links are shared. The matrix C (created by the BS according to the routing table) determines the links between the nodes:

$$C = \begin{bmatrix} c_{1,1} & c_{1,2} & . & . & c_{1,m} \\ c_{2,1} & c_{2,2} & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ c_{m,1} & . & . & . & c_{m,m} \end{bmatrix} \tag{6.9}$$

   where C is a $m \times m$ matrix, m is the number of the nodes and $c_{i,j}$ is the status of the nodes in the path;

$$c_{i,j} = \begin{cases} 1 & n_j \in P_{i,0} \\ 0 & \text{others} \end{cases} \tag{6.10}$$

   where $P_{i,0}$ denotes to the minimum cost path between node n_i and BS. The sum of all the entries of each row is the hop count for that node.

$$h_i = \sum_{j=1}^{m} c_{i,j} \tag{6.11}$$

3. **Allocate the time slots**: To allocate the time slots, it is necessary to find the branch from BS that has the largest number of nodes. The first node of each branch is one hop away. To find the number of nodes that shares a given node that is one hop away, it is sufficient to count all the elements that are set to 1 in the correspondent column, so

$$\text{number of the nodes sharing } n_i \ = \sum_{k=1}^{m} c_{k,i} \tag{6.12}$$

   Then the time-slot assignment starts, one by one, as described before. At the end, a linear arrangement of nodes may remain. In this case, the procedure described before for line arrangements is applied.

4. **Sending the slot assignment information to the SNs**: BS broadcasts a message that specifies the duration of each time slot based on th e allocated packet length. In addition, that message contains information items of each SN: assigned time slot, hop count and number of *RTSnd* messages in the time slot.

To clarify the above steps, assume that the BS in Fig. 6.1 builds the C matrix. The 19×19

matrix is given by

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & . & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & . & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & . & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & . & 0 & 0 & 0 \\ . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & . & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & . & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & . & 0 & 1 & 1 \end{bmatrix} \tag{6.13}$$

and the corresponding hop counts are: $h_1 = 1$, $h_2 = 2$, $h_3 = 2$, $h_4 = 3, ...,$ $h_{17} = 2$, $h_{18} = 1$, $h_{19} = 2$.

The first column shows that n_1 is one hop away from BS and that n_2, n_3, n_4 have n_1 in their first column, while n_{17} and n_{19} have n_{18} in their 18th column. The time slot assignment is then performed as explained earlier. The remained line is n_1, n_3 and n_4 with maximum number of *RTSnd* equal to $\lfloor \frac{3}{2} \rfloor + 1 = 2$ $(l = 3)$.

The network setup performed by the SRMCF protocol has two phases (Section 3.1.2). The 4th step of the time slot assignment algorithm requires an additional step when the network is set up to work in a TDM scheme. So, for TDM, the setup steps are: 1) Determination of node cost value; 2) routing table creation; 3) time slot assignment.

6.1.4 End-to-end delay

Assume that a node with l hops away from BS sends a packet. Regardless of utilized scheduling method, l RTS messages are needed to generate the path. So, the end-to-end delay, which is the time difference between the moment that the packet is placed in the buffer and the moment when it has been completely received at the BS will be

$$T_l = lT_r + T_c + T_P \tag{6.14}$$

In Chapter 3 it was described that for the case of random networks, with N nodes, both the maximum and the average number of hops in paths between nodes increase as $\Theta(\sqrt{N})$ [AER08, SMSR02]. In this case, the average hop count can be defined as $h\sqrt{N}$ where h is a positive value that depends on the network topology. From Eq. (6.14) the average end-to-end delay is

$$T \approx h\sqrt{N}T_r + T_c + T_P \tag{6.15}$$

Usually, the length of the packets is much bigger than the length of the RTS and CTS messages and T contains only one packet duration. Hence, it is expected that the end-to-end delay will be much smaller than the time required for packet switching, which is dominated by the delays accumulated in the intermediate buffers (Eq. (4.39)).

6.1.5 Bit Error Probability for Circuit Switching

In Section 4.2.1.1 the node-to-node error probability was calculated. Because circuit switching uses a different communication mechanism in which the signals propagate from the sender up to the destination without regeneration, the error probability is not the same as for packet switching. To calculate the bit error probability consider a node h hops away and white noise with σ_i^2 for the ith link. In a system with a linear transfer function $H(f)$, the power spectral density of AWGN is $\frac{N_o}{2}|H(f)|^2$ with total power [CCR02]

$$\sigma^2 = \int_{-\infty}^{\infty} \frac{N_o}{2}|H(f)|^2 df \tag{6.16}$$

For a multi-hop system in which each intermediate node just repeats the received signal in the output port, the total AWGN power is the sum of all node-to-node AWGN power

$$\sigma_t^2 = \sum_{i=1}^{h} \sigma_i^2. \tag{6.17}$$

Then the error probability $P_{ec,h}$ can be obtained by using Eq. (6.16) in Eq. (4.18), producing

$$P_{ec,h} = Q\left(\frac{\Delta V}{2\sigma_t}\right) = Q\left(\frac{\Delta V}{2\sqrt{\sum_{i=1}^{h}\sigma_i^2}}\right) \tag{6.18}$$

For close nodes located in the same environment, it can be assumed that all of them have the same noise variance, so that

$$p_{ec,h} = Q\left(\frac{\Delta V}{2\sqrt{h}\sigma_t}\right). \tag{6.19}$$

So, the error probability increases exponentially with the square root of hop count.

In packet switching, each packet is regenerated at the next node, buffered and then retransmitted to the following node. So, the probability of successful transfer over a path with h hops is

$$p_{sp,h} = (1-p_{e,1})(1-p_{e,2})...(1-p_{e,h}) = \prod_{i=1}^{h}(1-p_{e,i}). \tag{6.20}$$

Assuming that all probabilities are equal ($p_{e,i} = p_e$) gives

$$p_{sp,h} = (1-p_e)^h, p_e \ll 1 \Rightarrow p_{sp,h} \approx 1 - hp_e \tag{6.21}$$

Then the error probability $p_{ep,h}$ experienced in case of packet switching is

$$p_{ep,h} = hp_e = hQ\left(\frac{\Delta V}{2\sigma_t}\right) \tag{6.22}$$

which is a linear increase of the error probability with h. Comparing Eq. (6.19) and Eq. (6.22) it can be concluded that the error probability for packet switching is smaller than for circuit switch-

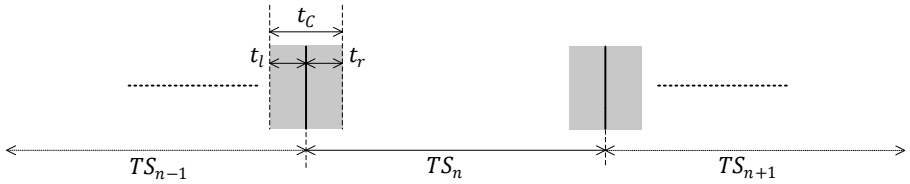


Figure 6.5: Time slots and the alignment error due to t_C.

ing. However, according to the experimental results shown later in the Section 5.1.4.3, in wearable systems with small size wire connections between nodes (supplied by batteries), the noise density is very small and the total error probability is acceptable in both cases.

6.1.6 Synchronization of Time Slots

Performance of a TDM base network significantly depends on the synchronization and time slot alignment. So, a synchronization method is needed to provide global synchronization and avoid overlapping of time slots. Consider a synchronization method in which the BS is the time reference and the global time skew is less than t_C. Figure 6.5 shows the time slots with alignment inaccuracy (highlighted in gray) due to synchronization. In the figure, t_r and t_l denote time deviations of the slot start time in relation to the time reference and

$$t_C = t_r + t_l \tag{6.23}$$

As before, T_g is the time gap between the time slots. In fact, T_g is the sum of the time gap at the beginning of a time slot and another gap at the end

$$T_g = T_{gr} + T_{gl}. \tag{6.24}$$

Any changes in the edges of slots that occur inside the gap do not affect communication. Since TDM is based on synchronization, time slot alignment accuracy will match the synchronization skew t_C. Then T_g has to be large enough to cover any changes in time slots due to synchronization skew.

$$\{T_{gr} > t_r,\ T_{gl} > t_l\} \quad \Rightarrow \quad T_g > t_C. \tag{6.25}$$

The value of t_C depends on the accuracy of the synchronization method. The IC presented in this chapter contains a highly precise time synchronization module that is able to provide a global synchronization in the range of sub-microsecond for multi-hop connections. Given the importance and complexity involved, the synchronization protocol and the hardware that implements the protocol in the MAC layer are explained in detail in the Section 5.2.

6.1.7 Hybrid switching

As referred earlier, the wearable network designed for this work supports node-to-BS, BS-to-node, node-to-node (just neighbors) and also broadcasting messages originated by any node. When

the network is set to work over packet switching, any kind of aforementioned communication performs normally, but in case of circuit switching different rules must apply. The main issue is that circuit switching applies to the node-to-BS case, but other kinds use packet switching. So, the network has to work in such a hybrid mode to support both switching modes without or at least with minimum system performance degradation.

Considering that the main communication is in the direction of node-to-BS which is based on circuit switching, packet switching has to obey the time scheduling of circuit switching discussed in the previous sections. Suppose that in the network of Fig. 6.1, n_4 is sending directly a packet to the BS over a circuit. At the same time, the other connections are free to perform node-to-node packet delivery, e.g., from n_9 to n_{10} or any other node except the links (not nodes) involved in the circuit path between n_4 and BS. It should be noted that because of the multi-task ability of the SNs, any node that is involved in circuit path also is able to contribute with a neighbor node, e.g. n_1 can communicate with n_2 while handling communication between n_4 and BS over the circuit path.

In general, it can be said that in each time slot, any link which is not involved in the path generated in that time slot is available to perform node-to-node packet delivery. Let L denote the set of all the links in wearable network.

$$L = \{l_{i,j} | n_i, n_j \in M, i \neq j\} \tag{6.26}$$

where $l_{i,j}$ denotes to the link between nodes n_i and n_j. Let P_k denote the path between n_k and BS with minimum cost forwarding. The set of available links for performing node-to-node packet delivery will be

$$L_p = \{l_{i,j} \in L | l_{i,j} \notin P_k\}. \tag{6.27}$$

The set L_p contains a significant number of network links. So, any kind of packet switching can be done in the network with a significant effect on circuit switching.

6.2 Integrated Circuit for Sensor Node Communication

The block diagram of the complete integrated communication circuit is shown in Fig. 6.6. This circuit inherits a large part of its functionality from the first prototype. The new features include the implementation of a hybrid packet/circuit routing protocol in hardware, a highly precise time synchronization circuit that generates a globally synchronized clock signal and accurate time stamps for the acquired data. In the first prototype, the time synchronization algorithm was implemented at the application level and run on the microcontroller. Each IC has one CDR circuit per port. Therefore, a very small fully-digital circuit CDR was designed, which is explained in detail in Section 5.1. The IC requires several of such circuits because it is designed to be able to run several communication tasks simultaneously.

In the first version, routing was completely implemented in software. To reduce packet serving time and power consumption, the new version implements routing from SNs to BS in hardware.

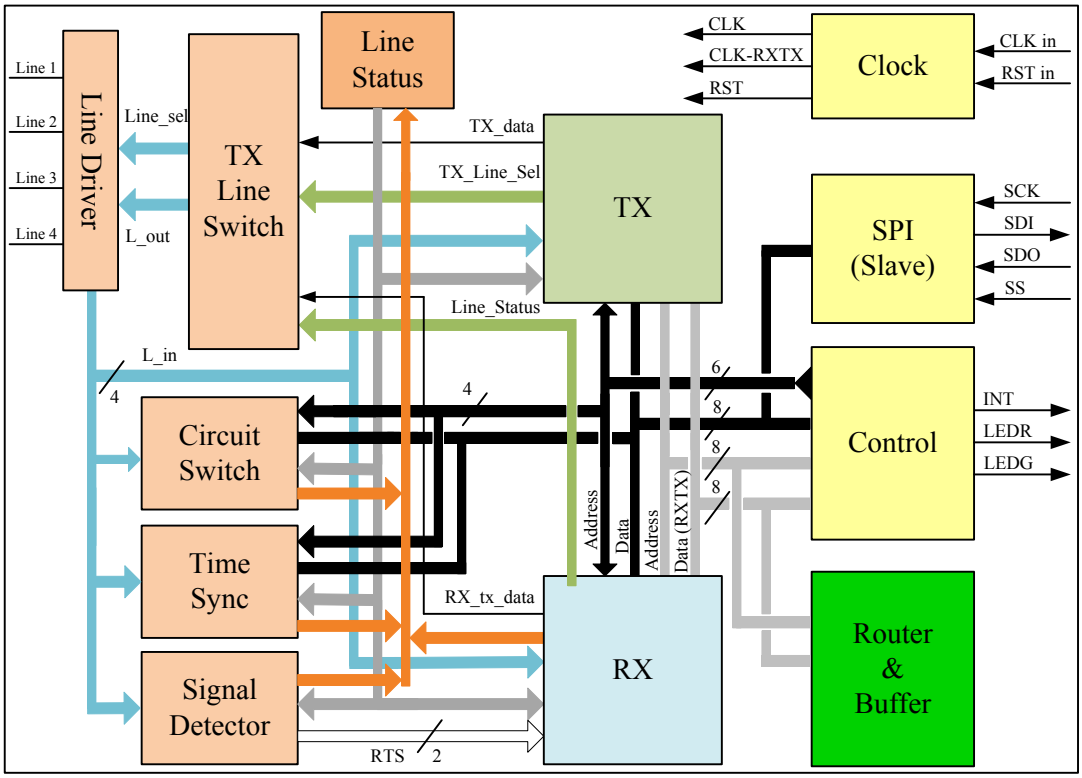


Figure 6.6: Network integrated circuit.

Such an implementation significantly reduces power consumption and end-to-end delay, as explained in the next subsection.

The power consumption in sleep mode also is improved. During idle time, the internal *Clock* module puts all the other modules except *Time sync* in sleep mode (disabled clock). Reception of any preamble signals or any SPI communication wakes up the system. At the end of the task, the system immediately turns to sleep mode for power saving. Further power savings can be obtained if time synchronization is not activated. In the next subsections, the functionality of each submodule is explained, except for those related to time and clock synchronization, which are described in detail in Chapter 5.

6.2.1 Router and Buffer Module

The buffer memory is shared with the microcontroller, which is able to access the packets stored in it. To process a received packet, the microcontroller reads seven bytes of the header of the segment which contains the packet. The measured packet serving time with the first prototype was 184 µs@16 MHz, which is the main factor impacting on the end-to-end delay and therefore becomes a performance bottleneck. That value is mainly due to the fact that, in the first prototype, the routing algorithm runs on the microcontroller. The reasons for the delay are the following:

1. Although SPI is a high-speed serial port (configured to run at 4 Mbps), reading or writing the header of data from or to the buffer takes a significant amount of time because of the time

interval between each data-byte transmission. The delay is then due to running functions on the microcontroller as a response to an SPI interrupt event.

2. The microcontroller has to decapsulate the information to determine the kind of packet that is being processed. Supporting several kinds of packets (including node-to-base, base-to-node-broadcast and node-to-node) and calling the appropriate function for further operation also takes some time.

3. When the microcontroller, after processing the header of the packet, recognizes that it must be routed to another node, then the microcontroller changes some information in the header and rewrites the header to the buffer. Writing the same number of bytes that were previously read doubles the SPI delay.

In sensor networks the data flow is mainly in the direction from SNs to the BS. In fact, the BS generates and sends controlling messages, which are shorter than data packets. To improve the routing performance of the nodes in both directions, SRMCF protocol is used as discussed before. So, in this version the improvement of routing is focused on the packets generated by SNs. For that, a router module was added to the circuit, provided with direct access to the buffer. This module starts to serve the packets as they are placed in the buffer, either received by the *RX* module or generated by the node itself. The router is connected to *data bus* and reads or writes data from the buffer like the other modules. The routing of packets in this way avoids involving the microcontroller in the routing process for the most frequent case, decreasing both power consumption and end-to-end delay significantly.

The operation of the *Router* module is shown as a flow chart in Fig. 6.7 and can be explained as follows:

1. When a packet is placed in the buffer (either because it was received or because it was written by the microcontroller), the routing module starts to serve the packet. The router will detect the presence of the packet because the reading and writing pointers of the buffer segments will not be equal. The first byte of each buffer segment determines the status of the saved packet (cf. Section 4.3.2.3). So, the router only examines the *Status* field of the packet.

2. If the packet is received from another node, the *Status* field will be checked to determine the destination of the packet, which can be node-to-node or node-to-base. If it is a node-to-node packet, then the address of the segment containing the packet will be copied to the *io_addr* pointer, which will be used by the microcontroller to read the contents of the buffer. An interrupt signal generated by the *IO_Control* module commands the microcontroller to continue by reading and processing the packet. It should be noted that node-to-node covers all the routing types except node-to-base.

   If the received packet is node-to-base, then it should be routed to the BS by the *TX* module. For that, the router first checks the status of *TX*. If *TX* is not free, the module waits until

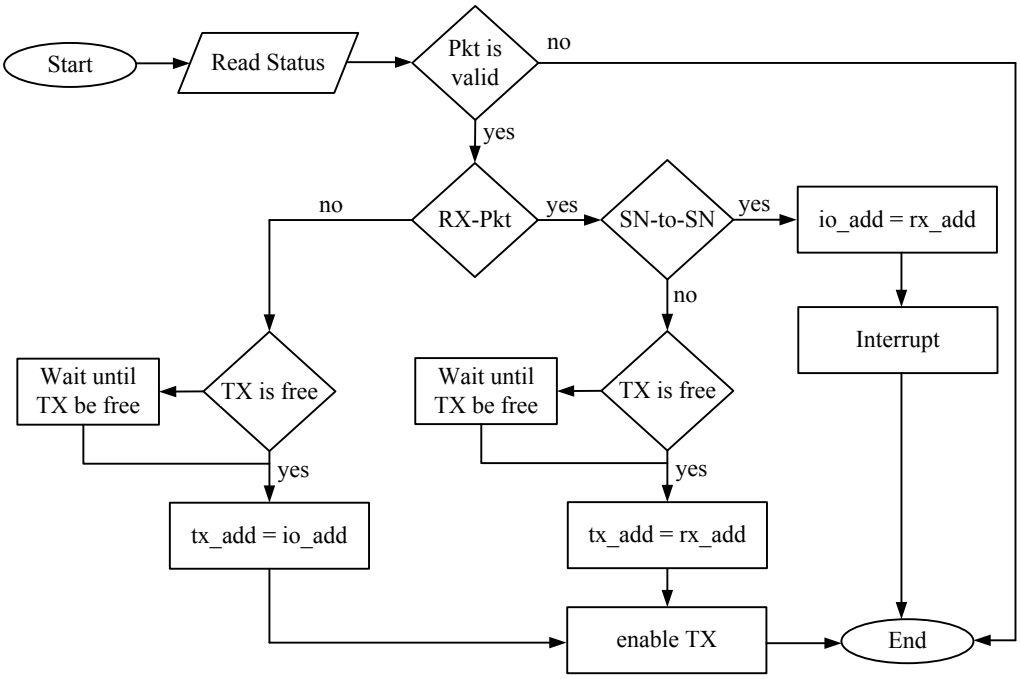


Figure 6.7: Flowchart describing the routing process implemented by the *Router* hardware module.

the end of communication, then copies the address of the segment to *tx_addr* (*tx_addr* = *rx_addr*) and enables *TX*. After that, *TX* tries to establish a communication with the *near-node* and continues with the sending process. The router will be free to serve other packets immediately after calling *TX*.

3. When the packet is generated by the microcontroller, then it has to be sent anyway. The process of checking *TX* and sending is the same as for received packets except that the address of the sender is *tx_addr* = *io_addr*.

This routing method decreases routing operation time and packet serving delay significantly and improves system performance. In addition, avoiding the involvement of the microcontroller in the routing process decreases power consumption. In fact, the intermediate nodes route the packets to the BS independently form the microcontroller. The quantitative results obtained with the circuit, which are described at the end of this chapter, confirm the advantages of this approach.

6.2.2 Transmitter Module

The structure of the *TX module* is similar to the one used in the previous prototype but with some new functionality. The main differences are:

1. As described in the previous chapter, a bit stuffing scheme has been applied to the data stream to ensure that a long string of zeros does not cause the CDR to be out of synchronization and make a wrong data recovery decision. For that, each group of 8-bit is followed

by one "1". Based on the clock drift of the quartz crystal which is used to stabilize the clock oscillator and the signal quality observed in practice, the platform is able to work with fewer synchronization bits, such as 16b/17b or even higher. To evaluate the impact of the different lengths of zero strings on synchronization, the ability of setting the sync coding bits to work in different modes was added to the circuit. The impact of the synchronization bits on clock recovery is analyzed and evaluated in Chapter 5.

2. The *TX* module is always responsible for starting packet transmission to the BS or to neighbor nodes. The type of switching depends on the destination, which can be BS (node-to-base) or a neighbor node (node-to-node) and the *TX* module must be able to the handle both of them. Depending on the switching mode, this module is able to generate *RTSnd* and *RTSnn* messages for circuit and packet modes, respectively. The *Encoder* submodule is responsible for generating the RTS signals. For the *RTSnn* mode, it generates the preamble signals followed by the *RTSnn* code, but for *RTSnd Encoder* also adds *hopCount*, whose initial value is one plus the CRC3 checksum (Fig. 6.8).

6.2.3 Receiver Module

The receiver module also inherits the main functionality from the first prototype. In the previous circuit, absence of the synchronization bit indicates the end of the packet. Because the new version supports multiple coding formats as explained earlier, that method cannot be used anymore. To control the communication and determine the end of packet, a counter is loaded with the packet length at the beginning of packet processing and starts to count down as the receiver gets the data. When the value of the counter reaches zero, *RX* finalizes the reception. After a successful packet reception, *RX* changes the *Status* field of the buffer segment that includes the packet to inform the router that a valid packet was received.

6.2.4 Signal Detector

In the first prototype, only one RTS message type is used, because only one kind of MAC frame is supported. However, in the new version MAC frames can be of one of 3 different kinds: packet switch, circuit switch and time synchronization. So, an RTS message carries more information. The module *Signal Detector* performs a pre-detection and then selects to module responsible for handling the message, which can be *RX* for packet switching, the *CSW* module for circuit switching and *Time Sync* for timing messages. The three types of RTS are shown in Fig. 6.8. They are:

1. *RTSnn*: This message is used for establishing communication over node-to-node packet switching and can be used for all types of communication: broadcasting, node-to-node, base-to-node or even node-to-base. Functionally, the *RTSnn* message is the same as the RTS message used in the first prototype.

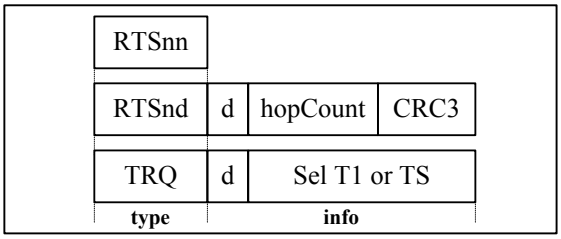


Figure 6.8: The three different types of RTS messages required by the new routing protocol.

2. *RTSnd*: If a sender node wants to send a packet to the BS, then it may request the intermediate nodes to establish a circuit path by sending a *RTSnd* message. This message also carries the hop count (field *hopCount*) to inform the nodes about the distance from the sender node which is used for limiting the length of the circuit (hybrid mode). The included information is protected by a CRC-3 checksum.

3. *TRQ*: The circuit contains a time synchronization module which is based on one-way, master-to-slave message exchange. To avoid all deterministic delay sources and increase the accuracy of synchronization, the circuit has been implemented in the MAC layer. The master node replies with a timing message whenever a slave node sends the request message *TRQ*. The master replies to the *TRQ* message with its time information. The details and analysis of the synchronization protocol and corresponding circuit implementation are discussed in Chapter 5.

Figure 6.9 shows the *Signal detector*, which is composed of *RTS Detectors*, *Receiver Sel* and *RST Signal gen* submodules. For the handling of concurrent requests, each port is equipped with its own *RTS Detector*. For example, while receiving a packet in one port, a time synchronization request (*TRQ*) may appear in another port.

At the end of a communication (packet reception, circuit switching or time message exchange), the submodule *RST Signal gen* generates internal reset signals to release the involved *RTS Detector*. For example, if *RX* is receiving a packet from port 1, then at the end of reception the signal *RX Busy* changes to low (deactivated) and *RST Signal gen* generates a pulse on *RX-RST* to release *RTS Detector 1*.

The *Receiver Sel* module generates enabling signals to select the responsible receiver. This module contains a ring counter that prevents the overlapping of requests. For example, if two *RTSnn* requests are received simultaneously from ports 2 and 3, then the *RX* module will be able to receive only from ports 2 or 3. In this situation *Receiver Sel* selects port 2 or 3 and release the other one.

Two line bus are used to share the recovered clock *CLKRX* and data *RXData* between detected *RTS Detector* and the receiver modules. So, the receiver module do not need to have a *CDR* circuit. Bus *RTS* is used by the *RX* module to handle the hybrid mode.

The *RTS Detector* module can be seen in Fig. 6.10. In idle mode, the status line indicates whether the line is free or occupied (either by transmitter or time synchronization module) and

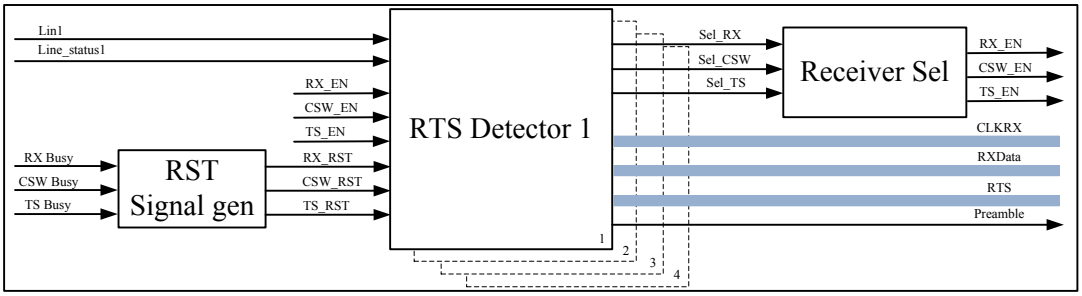


Figure 6.9: Signal detector module for tracking line activity.

able to receive signals or not. The first step of reception of a RTS message happens when the submodule *Predecoder* detects the preamble signals. This signal is used to wake up the receiver and is connected to the *Clock* module. It is worth noting that the *Clock* module stops sending all clock signals to the modules in sleep mode and needs to be waken up by preamble signals from the receiver modules or request signals from the other parts of system. In any case, to generate the preamble signal *Predecoder* does not need to run with system clock. For enabling the preamble signal *Predecoder* has to detect at least two consecutive transitions in the incoming signal.

After waking up the system, the *Clock Recovery* module generates a recovered *CLKRX* which is synchronized with the received signal and is used for data recovery. The operation of this module is explained and analyzed in Chapter 5.

After detecting the preamble signals, the *Predecoder* checks the RTS message. For example, if it receives an *RTSnn* message, it enables *Sel_RX_int* as a request to enable a *RX* module and then hands over communication to it. At this stage, *Select Receiver module* evaluates the availability of *RX* by checking the *RX_EN* signal. If the receiver is available, then the *CLKRX* and *RXData* lines will be enabled by the *Share CLKRX RXData*. This module operates as a switch to distribute the recovered clock *CLKRX* and data *RXData*. For *RTSnd* or *TRQ* messages, the *info* field of the message is decoded by the responsible module, not by *RTS detector*. As explained earlier, at the end of communication a reset signal releases the *RTS Detector*.

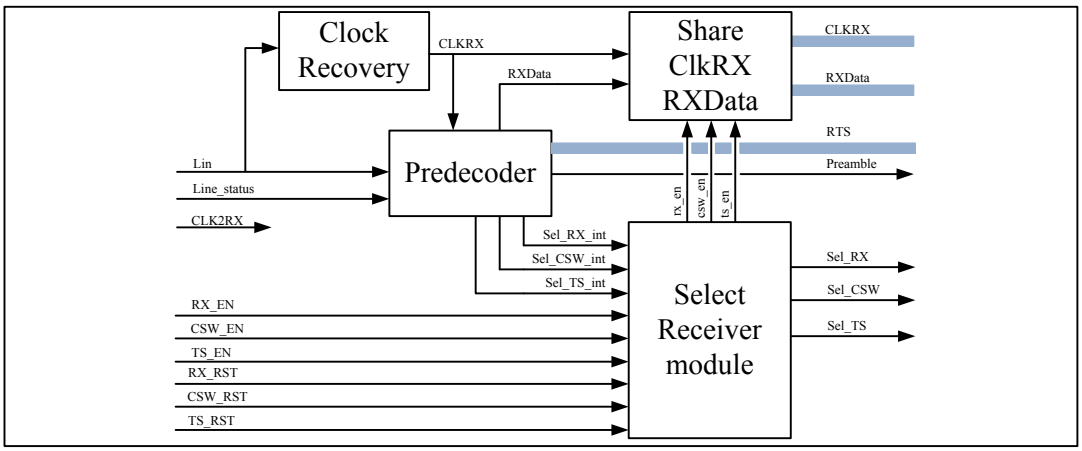


Figure 6.10: RTS detector module

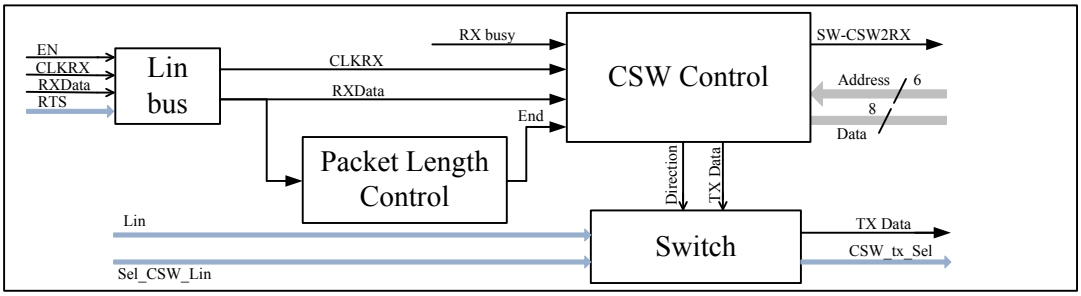


Figure 6.11: Organization of the circuit switching module

6.2.5 Circuit Switching Module

Participation of each node in circuit-based transmission is managed by the *Circuit SW* module. This module is responsible for establishing a path between an input and an output port, for monitoring the data transfer during communication, and for releasing the path at the end. Figure 6.11 shows the block diagram of the module. The *CSW Control* module controls all switching steps, generates *RTSnd* messages, evaluates the received *hopCount*, detects *CTS* messages, enables and changes the direction of the *Switch* module and switches to hybrid mode. The *Switch* module is a switch to establish a path between the selected ports.

To explain the circuit switching operation, consider a network of 3 sensors and BS connected as shown in Fig. 6.12. Assume that SN3 has generated or received a data packet and wants to send it to the BS via circuit switching. Figure 6.13 depicts the signals and status of all the nodes while communicating. It can be seen that some of the information appears on more than one line simultaneously. Sending a packet from SN3 to BS using circuit switching is done in 6 steps as follows:

1. *Step 1*: First SN3 sends a request message *RTSnd* to its near-node (SN2) with *hopCount* set to 1. In Fig. 6.13, the signals tagged with `Lyz` are the signals on the line connecting SNy to SNz; tags `SNx-x` represent the status of the ports, e.g., SN2-0 refers to port 0 of SN2. The low level (zero) represents that the node is transmitting and a high level represents receiving or listening status. So, in this step SN3-3 is low because SN3 is transmitting.

2. *Step 2*: The *Signal detector* of SN2 detects the *RTSnd* and then checks the availability of *Circuit switching*. If it is free (*CSW Busy* = 0), then *Signal detector* enables and forwards the rest of the message (*hopCount*+*CRC3*) to the *Circuit switching*. Then the *Circuit switching control* submodule receives the *hopCount* value, which is 1 at this step. If the port connected to the near-node of SN2 (i.e., SN1) is free, then it will send a waiting message (*w* signals in

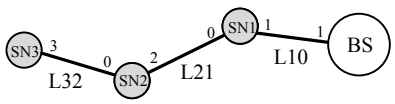


Figure 6.12: Example: a network circuit with 3 SNs and BS

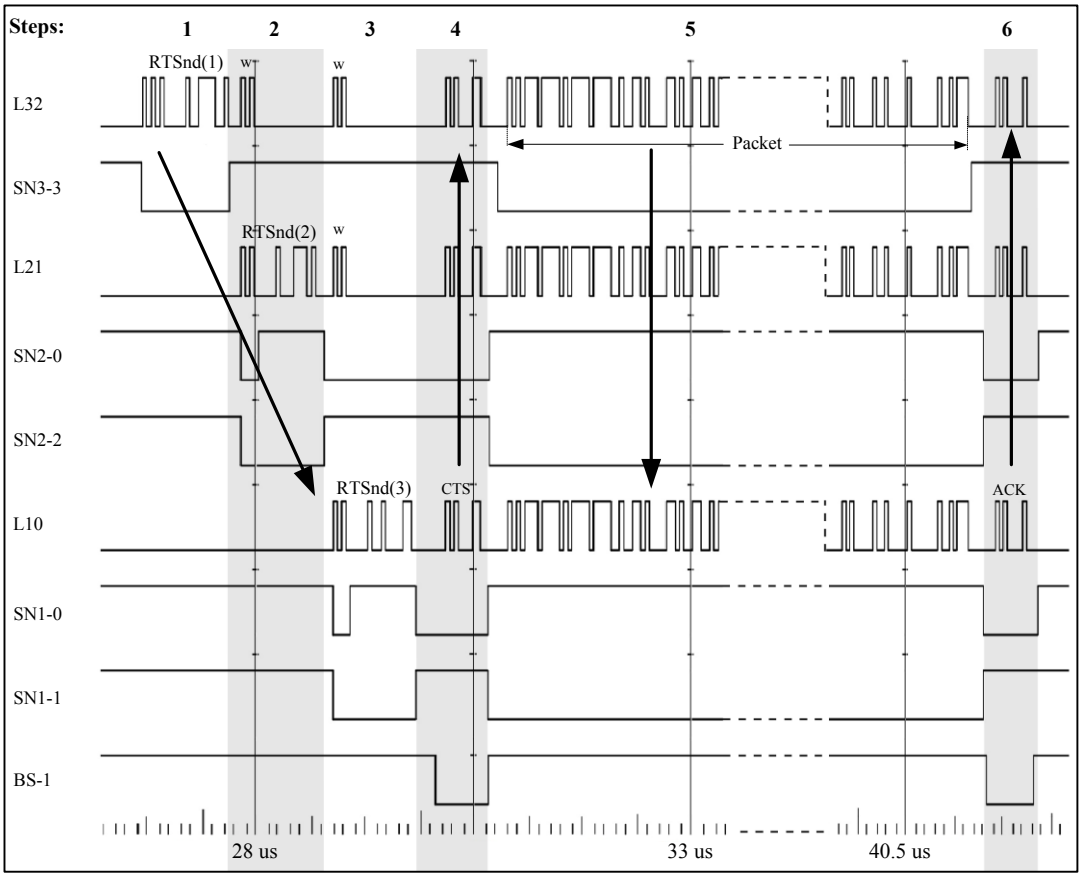


Figure 6.13: Example of packet delivery over a circuit involving two intermediate nodes

the figure) back to SN3 and concurrently a *RTSnd* message including *hopCount+1* to SN1. At the end of the message, SN2 changes the *switch* direction to pass all the signals coming from the line connected to the near-node.

3. *Step 3*: SN1 (as the near-node of SN2) follows the same process as described in step 2 by sending *RTSnd* to the BS and waiting signals to SN2. Because SN2 is passing all signals to SN3, this last node receives the *waiting* signal generated by SN1. The *hopCount* increases to 3 in this step.

4. *Step 4*: BS will reply to the *RTSnd* message with a *CTS* message. All the SNs on the path and the SN3 (as sender node) receive that message.

5. *Step 5*: Then SN1 and SN2 change the direction of the switch to let SN3 send the packet directly to the BS without any buffering. The second byte of the packet specifies the length of the whole packet. To find the end of the packet, submodule *Packet length Control* extracts the length information and loads it to a counter, which counts down as the packet passes on its way to the BS. The zero value of that counter determines the end of the packet.

6. *Step 6*: At the end of the packet, SN1 and SN2 again change the direction of their switch to let BS to send the ACK message to SN3.

Obviously, none of the intermediate nodes buffer the packet, which is the main advantage of the circuit switching to reduce power consumption and also end-to-end delay. Although circuit switching reduces the buffering and eliminates packet serving delay, long circuit paths may degrade network performance. To control path length, each node imposes two limitations on the number of hops:

1. *Maximum hopCount* (*maxh*): If *hopCount* in an SN reaches a maximum predefined value, then that node breaks the circuit without checking the status of its near-node, buffers the packet and sends it later to the BS. Such a network behavior implements packet delivery over hybrid circuit and packet switching. To receive the packet, the *CSW Control* module first checks the availability of *RX*, then activates the *SW-CSW2RX* line to let the *Signal detector* module enable the *RX* module. The *RX* will continue operating normally for packet reception.

2. *hopCount threshold* (*htr*): If *hopCount* is less than a predefined threshold value but the near-node is not ready to participate in circuit communication, then all previous nodes release the circuit and the sender must try later. If the *hopCount* is above the threshold, then the choice between buffering the packet or continuing the construction of the circuit depends on the status of the near-node (the next node on the circuit path).

The network performance and behavior depends on the *maxh* and *htr* values. In fact, it is possible to configure the nodes in such a way that communication can be done over packet, circuit

or a hybrid packet/circuit switching only by setting these two variables. For example, assume node n_{10} in Fig. 4.6 wants to send a packet to the BS using only packet switching. For that, *maxh* has to be 1. So for each packet, only a circuit path with one hop is allowed, which is actually normal packet switching. With $maxh = 1$ the value of htr is unimportant.

Now consider that both *maxh* and htr are set to 15 (or any number bigger than the hop count of n_{10}). In this case, all intermediate nodes try to establish a circuit path if possible; otherwise, they will discard the path. So, setting *htr* to a value greater than the largest hop distance in the network ensures that only circuit switching is used.

Now assume that $maxh = 4$ and $htr < 4$. In this case, the circuit path length cannot exceed 4, meaning that the communication between n_{10} and BS uses circuit switching between n_{10} and n_3 and packet switching between n_3 and BS. Depending on the htr and *maxh* values, packet delivery may use more combination of packet and circuit switching. So, by setting both *maxh* and htr, many combinations in hybrid mode are possible.

6.2.6 Time Synchronization Module

In the application layer, time stamping of the captured data for analyzing the data is essential for further post processing. The sensor includes a circuit to implement a time synchronization protocol in MAC layer which is based on master-to-slave one-way message exchange. This module is able to communicate and exchange the timing messages completely independent form the other parts. The theoretical and experimental results confirms that synchronization is in the range of a few nano-seconds, which is accurate enough to satisfy many BAN applications [DFT14b]. The circuit generates a synchronized clock *Clk-sync* available at one of the pins and a real time value for time stamping. The protocol and the circuit are explained and analyzed in Section 5.2.

6.2.7 Clocking

As a digital system, each module needs a clock signal. The system clock is also used to synchronize the data flow inside and between modules. On the other hand, each module may require different clock frequencies. For example, the communication data rate (determined by both *CLK2TX* and *CLK2RX*) can be equal or less than the system clock frequency. The *Clock* module shown in Fig. 6.14(a) is responsible for generating, controlling and distributing all clock signals.

The *Reset* submodule generates an internal synchronous reset signal. To reset the entire system, the *RST-pin* must be held high for at least one clock period. The *Generate* module contains some counters for generating several clocks:

1. *CLK*: This is the system clock with the same frequency of the signal on *CLK-pin* (external clock source) and drives all modules. Because this clock is used as a reference to synchronize the data flow in the system, a clock grid was used to distribute it to the modules with small clock skew.

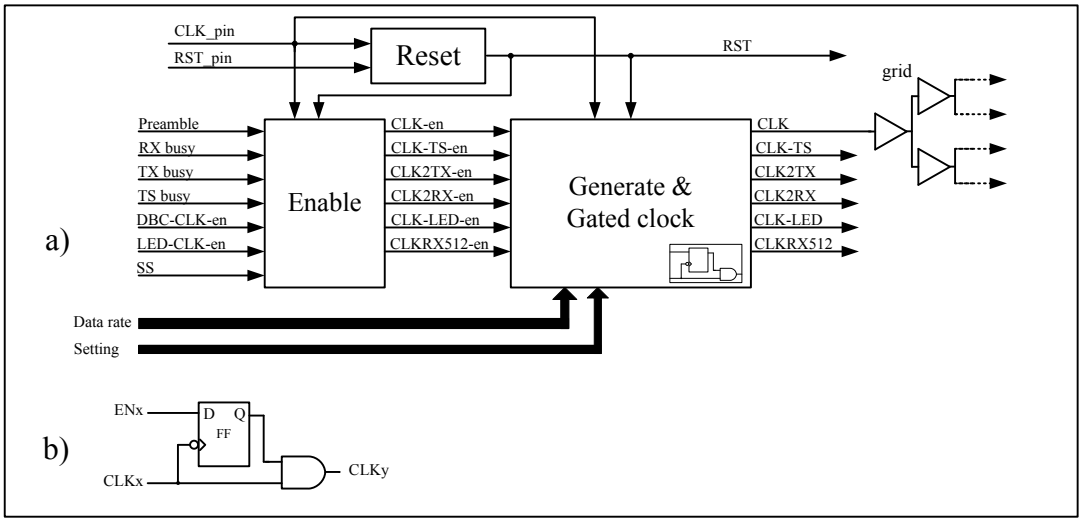


Figure 6.14: a) Organization of the *Clock* module, b) gated clock.

2. *CLK-TS*: This is the clock of the *Time-sync* module. In sleep mode, the *CLK* clock is not active, but the *Time-sync* module for generating synchronized time stamps and a globally synchronized *Clk-sync* signal still needs an active clock signal. Hence, *CLK-TS* provides a separate clock signal to drive the *Time-sync* in both sleep and active modes.

3. *CLK2RX* and *CLK2TX*: The communication IC supports multiple data rates, so the receiver and transmitter have their own clock signals *CLK2RX* and *CLK2TX*, respectively. The Data rate is programmable (via the SPI interface). This module generates the appropriate clock signals based on the selected data rates. The range of data rates goes from 1/2 to 1/256 of the system clock frequency.

4. *CLK-LED*: The circuit drives two LEDs to indicate the system status while running and communicating. These LEDs turn on and off with a period which is much lower than the period of the system clock. This module uses a counter to generate the *CLK-LED* clock.

5. *CLKRX512*: During normal operation of the circuit, unexpected conditions or failures may occur, possibly resulting in system malfunctions. To handle such a conditions, the modules of the system such as *RX* and *TX* contain a watchdog timer circuit driving by the *CLKRX512* clock. The clock period of this signal is 1/512 of *CLK2RX*.

As mentioned before, to reduce power consumption, the system can be put into sleep mode whenever there is no need for tasks to be executed. In sleep mode all the aforementioned clocks except *CLK-TS* will be disabled by the *Clock* module. To synchronously activate or deactivate all clocks, the *gated-clock* shown in Fig. 6.14(b) is used, which is based on activating the clock *CLKy* at the output by sampling the clock enable *ENx* signal at the negative edge of the clock *CLKx*. In this way, the active edge of all clock signals occurs at the same time. The *Enable* submodule generates all the enabling signals based on the tasks that must be performed. For example, if the node is receiving a packet, then *RX busy* enables *CLK*, *CLK2RX*, *CLKRX512* and *CLK-LED*.

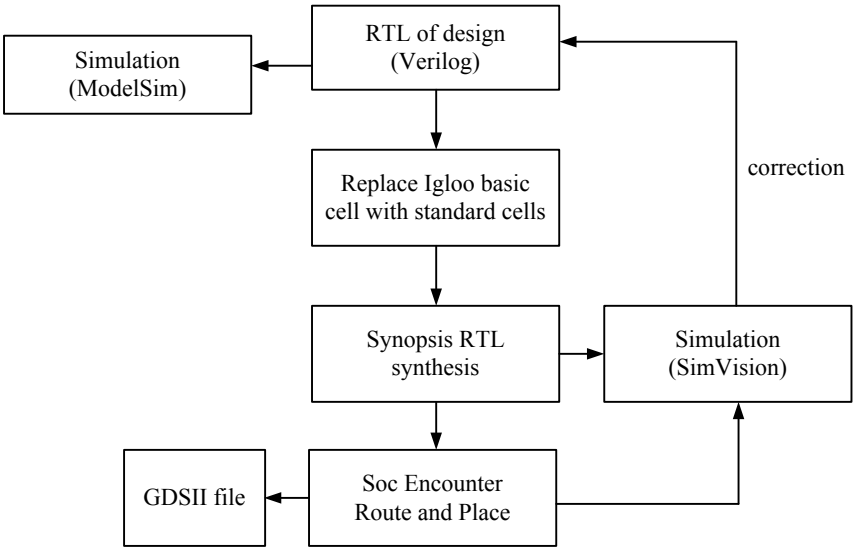


Figure 6.15: Design flow of the circuit for both ASIC and FPGA.

In sleep mode, two conditions may occur which require the system to change its status to active: reception of preamble bits from a neighbour SN or SPI port activity initiated by the microcontroller by pulling pin SS to low. After waking up, the system will stay in active mode as long as the requested task is proceeding.

6.3 ASIC Implementation Flow

The first step of the design flow is to create a Register Transfer Language (RTL) description of the system functionality in a Hardware Description Language (HDL) language, which is Verilog for this project. For the first prototype, the implementation target was a low-power FPGA from Actel but here the target is a standard-cell IC technology. Since an FPGA version can also be useful for wearable applications [DFT14a], the new version can be used for both platforms. In fact, except for the use of some Actel macros, the other functions are the same. and the corresponding RTL descriptions are implementable on both platforms. For that, the circuit has been developed with Libero IDE from Actel, a complete design environment for FPGAs. Then, a custom program was designed and used to replace the Actel macros with standard cells. In this way, two Verilog files are generated, one for FPGA and another one for the ASIC version. Figure 6.15 shows an overview of the design flow of the circuit.

Libero IDE uses the ModelSim tool for validation of the design. The ASIC version was also simulated with the NCSim simulator form Cadence through NClaunch, a graphical user interface that allows the management of large design projects and the configuration and launching of the Cadence simulation tools. NCLaunch uses the SimVision waveform viewer for showing the results and interacting with the circuit under simulation.

After validation, the Design Compiler tool from Synopsys was used to perform logic synthesis and technology mapping to the Austria Micro Systems (4-metal CMOS 0.35 µm) cell library. Since

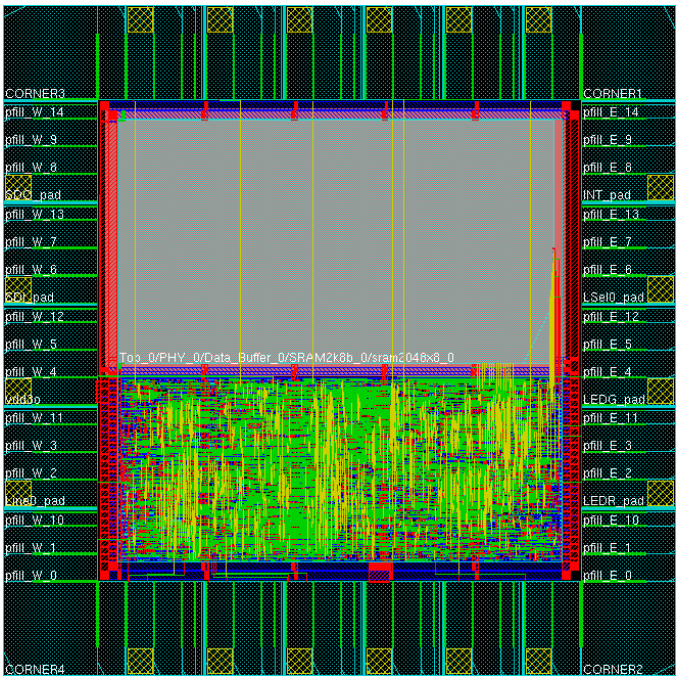


Figure 6.16: IC Layout; 4 metal CMOS 0.35 µm, core size = 2.53 mm^2, total size = 5.86 mm^2.

synthesis involves several steps, a script file detailing all the steps was created. This file also contains clock period specifications and constraints. Usually, depending on the synthesis results, it is necessary to go back to the RTL level, change the descriptions and compile again. Employing a script file makes the synthesis process simpler to manage. In addition to synthesizing the ASIC core, Design Compiler was used to create a structural, chip-level description of the ASIC that includes the input/output ports. The complete synthesized circuit is stored as a structural Verilog netlist, which can be used for simulation and for physical synthesis. In addition to the Verilog netlist, a file with synthesis constraints is produced.

Physical synthesis was carried out with Cadence Encounter. Figure 6.16 depicts the IC layout plotted by Encounter. Before running Encounter, the position and type of the I/O pads has to be defined. All the placement and routing steps in Soc Encounter were run from a script file provided by AMS. At the end of the process, a file containing fabrication information in the GDSII format is generated. For back-annotation to the simulator, a file with the calculated parasitic capacitors and resistors of the interconnects is also created (in the SPEF format). Additionally, a structural Verilog description of the circuit (including the clock tree) is also created. NCLaunch was used again for a final logic simulation with detailed delay information.

For validation of the descriptions in all the aforementioned steps, some testbenches have been designed. Since the ASIC is to be used in a sensor network, the testbench sets up a simple network with a BS and a few SNs. All operations of the ASIC are set and controlled by an external microcontroller via the SPI port. Hence, the nodes used in the testbench include a microcontroller, with an SPI port described in Verilog and connected to the ASIC. Each microcontroller communicates with the ASIC as in a real application. Based on the functionality under test, each microcontroller

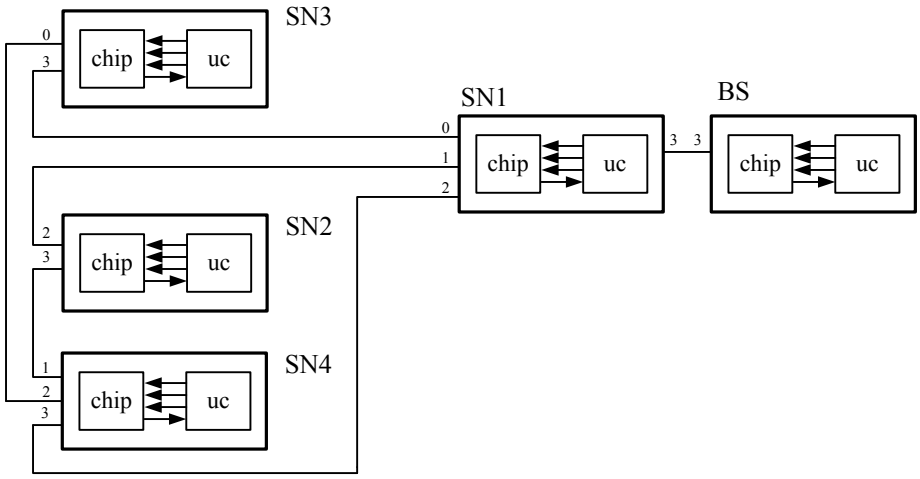


Figure 6.17: Example of a testbench with four SNs and BS.

is configured to generate the appropriate commands. Figure 6.17 shows the diagram of a testbench, including four SNs and a BS. Depending on the desired tests, the testbench can include more nodes in any connection pattern as in the real world.

6.4 Experimental Results

This section presents the experimental results obtained with the second sensor node prototype that includes the communication ASIC. The sensor node is shown in Fig. 6.18. The node was evaluated both with interconnection of normal copper wire and conductive yarns. Data rate, supply voltage, clock frequency, and other parameters of the IC are summarized in Table 6.1. Signals on the lines between the nodes have been observed and measured with a digital oscilloscope. The results on signal quality and performance of both clock and time synchronization are presented separately in Sections 5.1 and 5.2

Table 6.1: Characteristics of the communications IC

Parameter	**Value**
Technology	CMOS 0.35 µm
Power supply voltage	3.3 V
I/O	3.3 V
Clock frequency	up to 70 MHz
Data rate	up to 35 MHz
Die size	5.86 mm^2
Core size (with RAM)	2.53 mm^2
Internal memory	2 kB

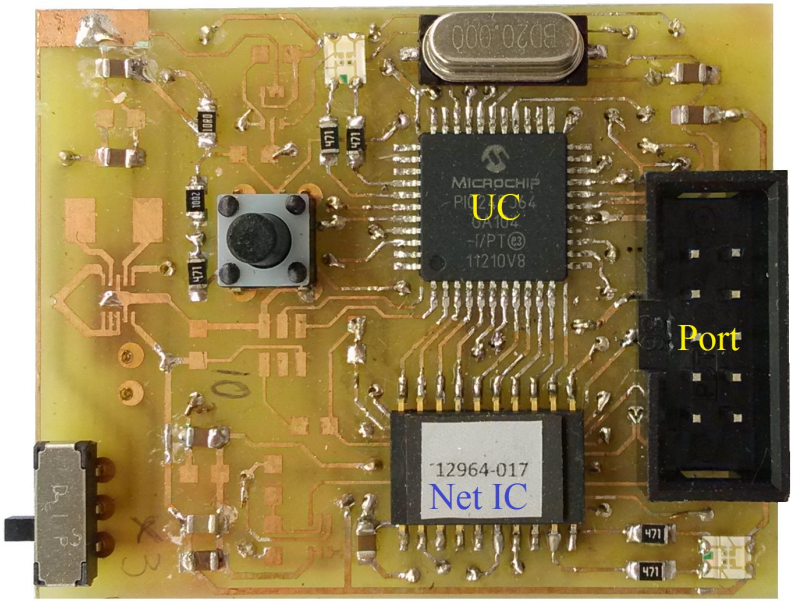


Figure 6.18: Photo of second sensor node prototype used for evaluating the ASIC.

The photograph of the fabricated ASIC is shown in Fig. 6.19. It has 20 pins including communication ports, SPI interface, power supply and test pins, and is packaged in a SOIC20 package, which is also shown in Fig. 6.19. To help evaluate the CDR, the recovered clock and the data line of port 0 are available on pins 14 and 15. These pins have been used only for evaluation; in normal operation they can be disabled by a software command. The ASIC has an internal module called *Line Driver* that drives the communication lines and controls their status (receiver or transmitter). The status of each line is controlled by a *Line-sel* signal. To monitor and evaluate the activity of the lines, line *Line-sel0* of port 0 is connected to pin 11.

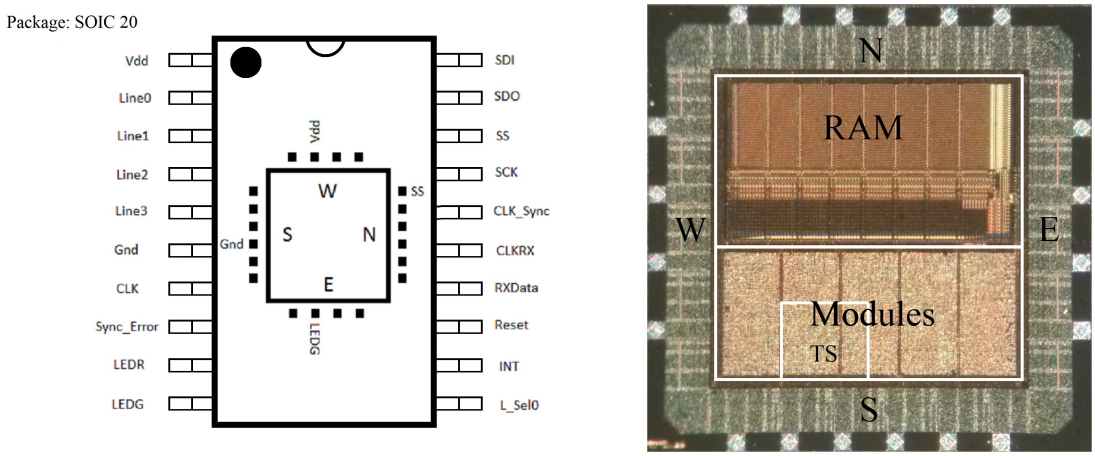


Figure 6.19: SOIC20 package and photograph of the integrated circuit.

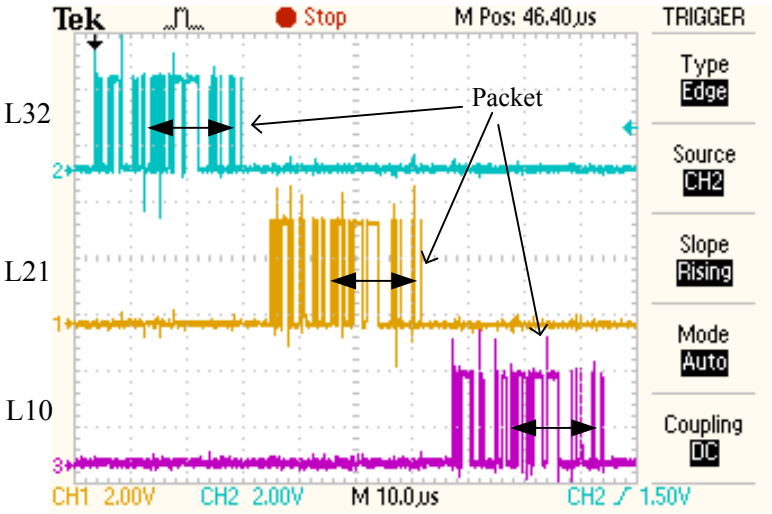


Figure 6.20: Delivering a packet generated by SN3 to BS only by packet switching.

6.4.1 Routing Operations

The different types of switching and routing operations have been explained in previous sections. This section evaluates the node-to-BS routing behavior and performance in all configurations (packet, circuit or hybrid mode) for the network of sensor nodes depicted in Fig. 6.12.

6.4.1.1 Routing Only by Packet Switching

Figure 6.20 depicts the signals during the transmission of a small data packet (4 bytes) generated by SN3 and sent to the BS via a path through intermediate nodes SN1 and SN2. Routing operation is independent from packet length and a small packet was selected to show the signals clearly. All signaling is the same as in the previous prototype. To configure the sensor nodes to act as a normal packet switch, the value of *Maximum hopCount* and *hopCount threshold* must be set to 1. All nodes are working at 16.383 MHz and the data rate is set to 4.096 Mbps. In each node, at the end of packet reception, the router starts to serve the packet and route it to the next node. The measured value for serving time is 5.3 µs, which is much smaller than the time measured for the previous prototype (184 µs), resulting in a significant improvement (34.7 times shorter) of the routing delay due to serving time. The measured end-to-end delay of sending the packet is 90.16 µs.

6.4.1.2 Routing Only by Circuit Switching

The signals generated while transmitting the same packet (as used in the previous experiment) but using pure circuit switching are shown in Fig. 6.21. SN3 starts establishing a circuit path by sending an *RTSnd* message at $t = 0.24\,\mu s$; transmission is completed at $t = 36.3\,\mu s$. So, in this case the end-to-end delay is 36.06 µs which is 0.4 of the time required by packet switching.

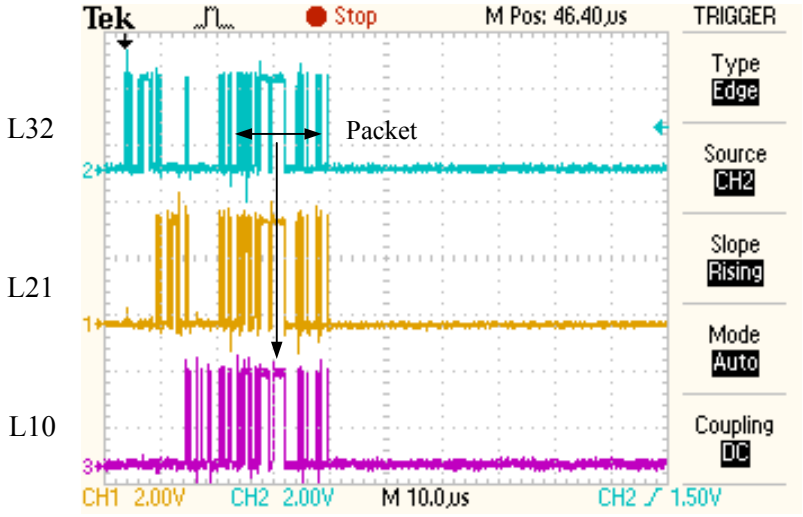


Figure 6.21: An example of pure circuit switching.

Figure 6.22 shows the message exchange required to establish the circuit path between SN3 and BS. As mentioned earlier, the *Line-sel* of port 0 is available on one of the pins and was used to measure the activity of port 0 of all intervening nodes, as shown in Fig. 6.23.

Node SN3 starts communication by sending *RTSnd + 1* to SN2 at $t = 0.24\,\mu s$ (cf. Fig. 6.22). Node SN2 receives the *RTS* message, simultaneously sends a *Wait* message back to SN3 and a *RTSnd + 2* message forward to SN1 at $t = 5.8\,\mu s$. SN1 starts communication with the BS at $t = 10.84\,\mu s$. At this moment, a circuit path has been established between BS and SN3 The path first is setup to transmit in the direction from BS to SN3. Therefore, SN3, SN1 and SN2 directly receive the *CTS* message from BS at $t = 17.12\,\mu s$. Then, SN2 and SN1 change the direction of the switch to allow SN3 to transfer the data packet starting at $t = 20.24\,\mu s$. The packet has 4 byte and takes $12.56\,\mu s$ to transmit, finishing at $t = 32.8\,\mu s$. At the end of the packet, SN2 and SN1 change the direction of their switches to transmit the *ACK* message from the BS. All nodes release the lines and finish the communication at $t = 36.28\,\mu s$.

6.4.1.3 Routing by Hybrid Switching

As explained before, a data packet can be sent to the BS by using both switching modes, thereby implementing a hybrid mode of transmission. To show the signals and network operation in this case, packet switching is used between SN3 and SN2; then, SN2 sends the buffered packet to the BS via circuit switching through SN1. Figure 6.24 shows the measured signals during hybrid switching. SN3 starts to send the packet at 0.24 µs, finishing at 2.87 µs. After processing the packet at 5.6 µs, SN2 sends it to the BS by establishing a circuit path via SN1, finishing at 63.6 µs. The measured end-to-end delay in this case is $t = (63.6 - 0.24)\mu s = 63.34\,\mu s$, which, as expected, is smaller than the time required by pure packet switching mode, but larger than the time required by pure circuit switching.

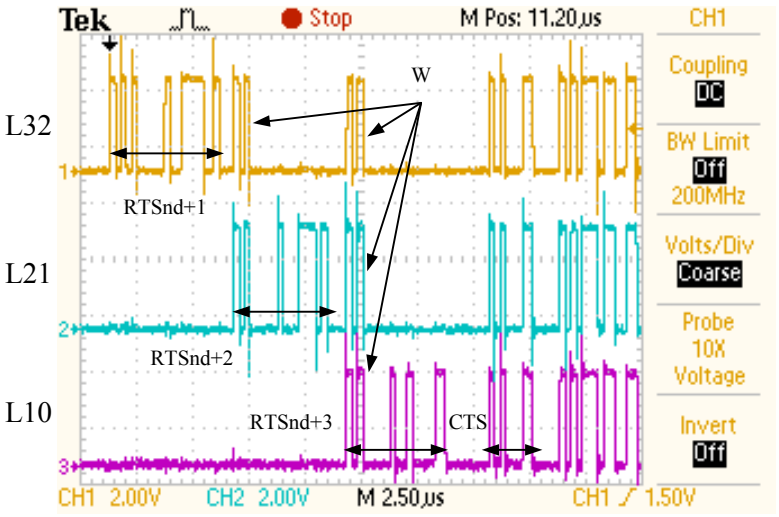


Figure 6.22: Details of message exchange to establish the circuit path between SN3 and BS.

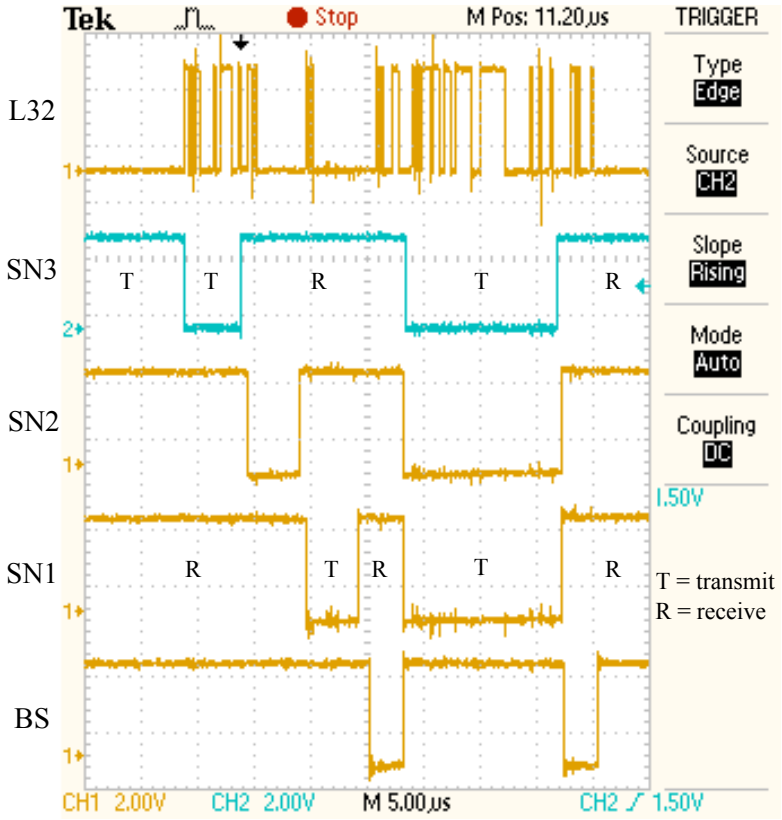


Figure 6.23: Status of the communication ports of each SN during circuit switching.

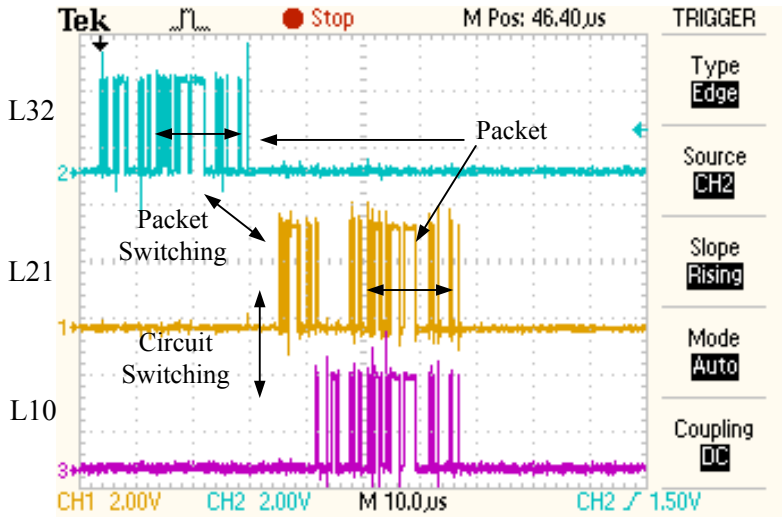


Figure 6.24: An example of hybrid switching.

6.4.2 Power consumption

In sleep mode, all internal clocks are disabled in order to save power. To separately evaluate the power consumption of receiver, transmitter and the other modules in sleep mode, the ability to disable the *CLK2RX*, *CLK2TX* and *CLK* signals has been added to the *Clock* module. In active mode all clock signals are permanently driving the respective modules. The results can be seen in Fig. 6.25 for both a 16.383 MHz and a 20 MHz system clock. The signs "-" and "+" represent sleep and active modes, respectively. When all clock signals are disabled, the current is at the the minimum values of 0.5 mA@16.383 MHz and 0.61 mA@20 MHz. When all clock signals are active, the systems require almost 2.64 time more current and therefore consume more power (supply voltage is constant).

Figure 6.26 depicts the measurement results of the power usage in terms of selected data rate with 20 MHz system clock. In sleep or when only *CLK* is active, the power consumption remains constant as expected. By increasing the data rate, the power consumption increases when either *CLK2RX* or *CLK2TX* are active. *CLK2TX* drives more elements than the *CLK2RX* and therefore has more impact on the supply current.

The power consumption of the ASIC when operating as a transmitter, receiver or participating in circuit switching is shown in Fig. 6.27. The power increases almost linearly with the data rate. In circuit switching mode, the circuit draws less current than in packet switching mode (either as transmitter or receiver). In packet switching mode each node receives and then sends the packet. Hence, power consumption for each packet in this mode requires the addition of both receiving plus transmitting power consumption, whereas in circuit switching, because of the absence of buffering, the power consumption is only due to the internal connection between the ports, which is smaller than the power required for packet switching. So, operating the ASIC in circuit switching mode decreases both power consumption and end-to-end delay.

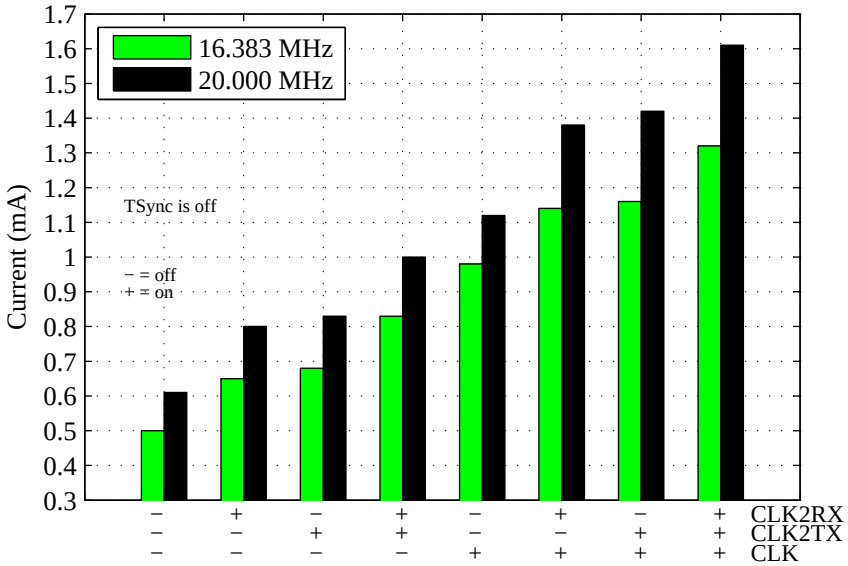


Figure 6.25: Measured ASIC supply current as a function of the activity of *CLK2RX*, *CLK2TX* and *CLK*.

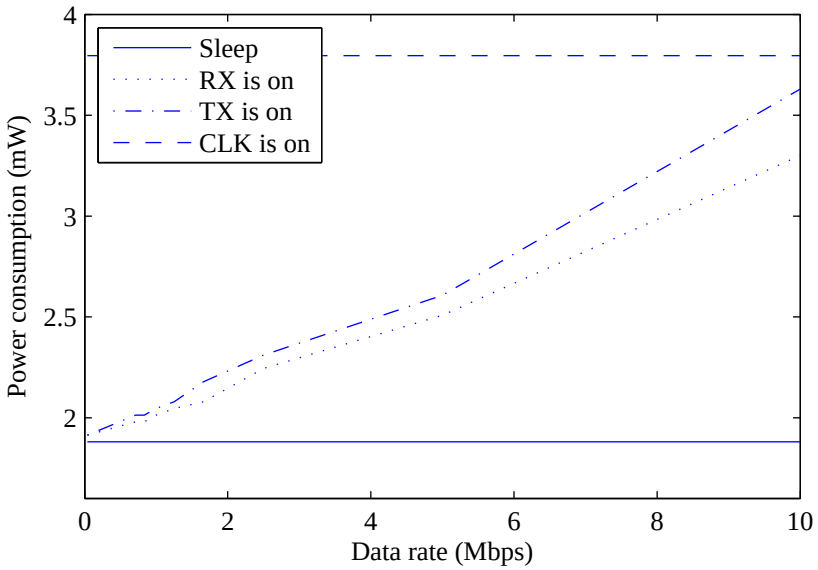


Figure 6.26: The measured power consumption in terms of data rate (VCC = 3.3 V).

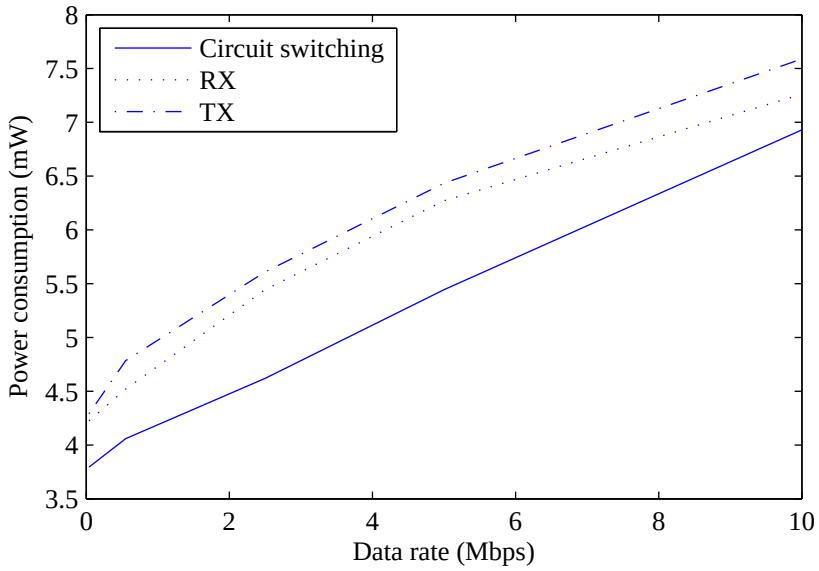


Figure 6.27: The measured power consumption in terms of data rate for different operation modes (VCC = 3.3 V).

6.4.3 Concurrent Multitasking

The ASIC is able to handle several concurrent tasks. For the evaluation of this characteristic, consider the network shown in Fig. 6.28. All ports of SN2 are used to communicate with BS and other SNs.

Figure 6.29 depicts the signals captured while performing three tasks on SN2. First of all, it establishes a circuit path between SN4 and BS. 24 µs after starting that communication, SN3 sends a packet. SN2 receives the packet from SN3 without any effect on the communication between SN4 and BS. The third communication is a time synchronization message exchange between SN5 and SN2. At this time, all ports of SN2 are occupied, but it can handle all tasks without any problem or any interruption.

6.4.4 Channel Utilization

The time interval between packets over the lines cannot be less than the serving time. In the second prototype, the routing process has been moved to the ASIC in order to provide much faster

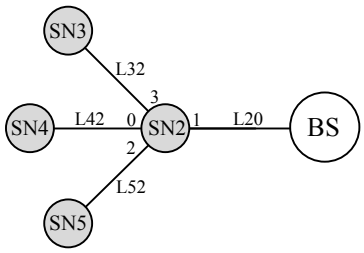


Figure 6.28: Network used for evaluation of communication multitasking.

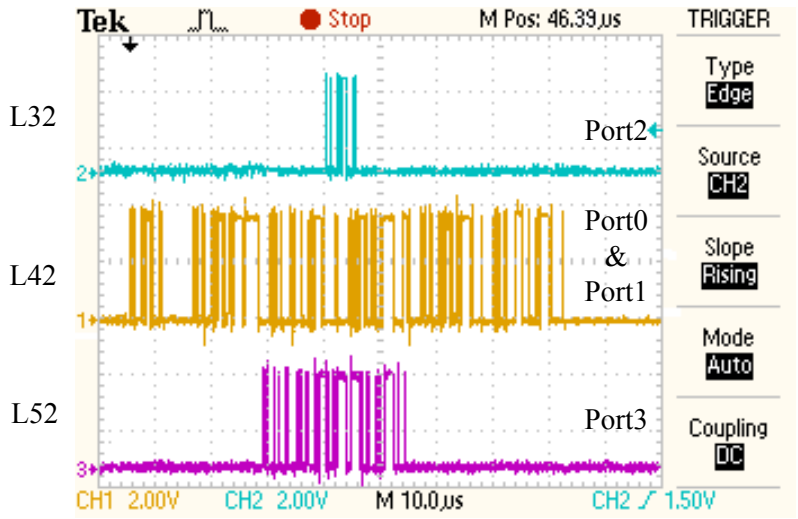


Figure 6.29: Signals on the data lines of SN2 while concurrently performing three communication tasks.

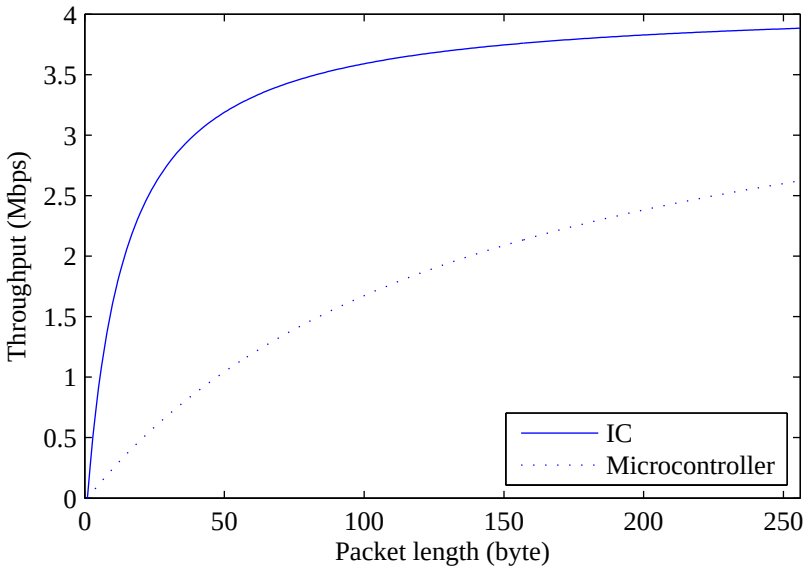


Figure 6.30: Channel utilization in MAC layer and comparison the results achieved by running the routing algorithm on the microcontroller.

packet processing (decapsulation) and faster routing as well. The channel utilization (or MAC layer throughput) achieved according to Eq. (4.34) is shown in Fig. 6.30. For comparison with the previous prototype, the data rate has been set to a quarter of the system clock (4.096 MHz). The maximum throughput is 95 %, which represents a 31 % improvement in comparison with doing the routing on the microcontroller.

6.4.5 Circuit Switching on Wearable Network

The performance of the network shown in Fig. 4.42 when using circuit switching was also evaluated. As mentioned, this network is used to acquired data from the lower limbs during locomotion. The set of the nodes is:

$$M = \{\text{BS}, \text{SN2}, \text{SN3}, \text{SN4}, \text{SN5}, \text{SN6}, \text{SN7}, \text{SN8}\} \tag{6.28}$$

The connection matrix for the network is:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{6.29}$$

and the hop counts are: $h_1 = 1,\ h_2 = 1,\ h_3 = 2,\ h_4 = 1,\ h_5 = 2,\ h_6 = 2,\ h_7 = 3$.

Three branches connect to BS; the branch that contains SN5 has the largest number of nodes. So, by starting with the branch that contains SN3, one finds that all nodes are assigned in the first iteration with one RTS message. SN2 also could have been selected first (instead of SN3), but the result would be the same. In this case the effect of the RTS message on throughput will be minimum (one RTS).

The total end-to-end delay in the aforementioned network for two data rates (4 Mbps and 8 Mbps) is depicted in Fig. 6.31. The delay increases linearly with packet length. Unlike the end-to-end results for packet switching shown in Fig. 4.47, in circuit switching the end-to-end delay is independent from the traffic. Circuit switching exhibits a delay of 0.58 ms@4 Mbps for a 250 B packet length. In comparison with packet switching at the same data rate (4 Mbps), it is 7 times less than the delay at 640 kbps and 3 times less at 20.8 kbps traffic.

Because in the second prototype data rate can be set to half the system clock, the 16 MHz system clock allow the system to operate at a data rate of 8 Mbps; for the same system clock frequency, the first prototype achieves a maximum data rate of 4 Mbps. The results shows that by doubling the data rate, the end-to-end delay decreases almost to half (for the same system clock frequency).

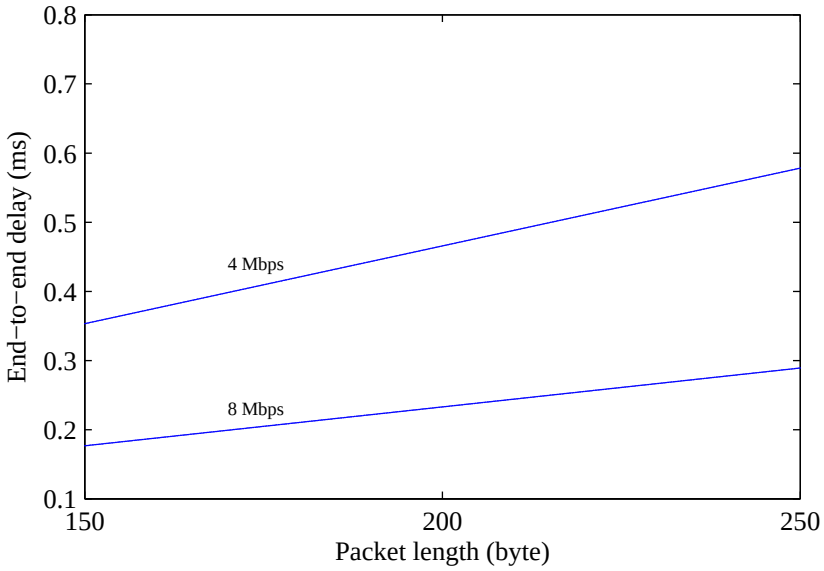


Figure 6.31: End-to-end delay as a function of packet length for two data rates.

Figures 6.32 and 6.33 show the throughput of the network as function of packet length for different amounts of total network traffic according to Eq. (6.2). A noticeable improvement in comparison with the previous version (which is shown in Fig. 4.48) is evident. For example, the first prototype exhibits 90 % throughput for 100 B packet length and 640 kB traffic, whereas in the second prototype, for the same throughput and packet length, data traffic can be increased to 3.33 Mbps, which is more than 5 times higher. To evaluate the effect of RTS on throughput, the results of both using $h_{max} = 3$ (Fig. 6.32) and the optimized value $h = 1$ are presented (Fig. 6.33). Because of the small scale of the network, the optimized method leads to a small improvement in throughput, but for larger networks the improvement will be more significant.

6.5 Conclusion

The structure of the communication ASIC and its functionality, which includes the on-chip implementation of a routing algorithm supporting both circuit and packet switching, time synchronization has been presented in this chapter. The design of an improved circuit started from the disadvantages of using only packet switching in the first prototype (as pointed out in Chapter 4). The IC was fabricated in a 0.35 µm technology and was thoroughly tested and evaluated. All oscilloscope measurements were obtained from a network of sensor nodes which included the IC. The ASIC worked correctly in all cases. The experimental results show that the circuit works up to 35 Mbps with less power consumption and higher throughput than the previous one. The results also confirm that in a wearable BAN consisting of nodes connected in a mesh topology, the power consumption of the intermediate nodes involved in constructing a circuit path and in packet handover for circuit switching is less than for packet switching. The IC contains several

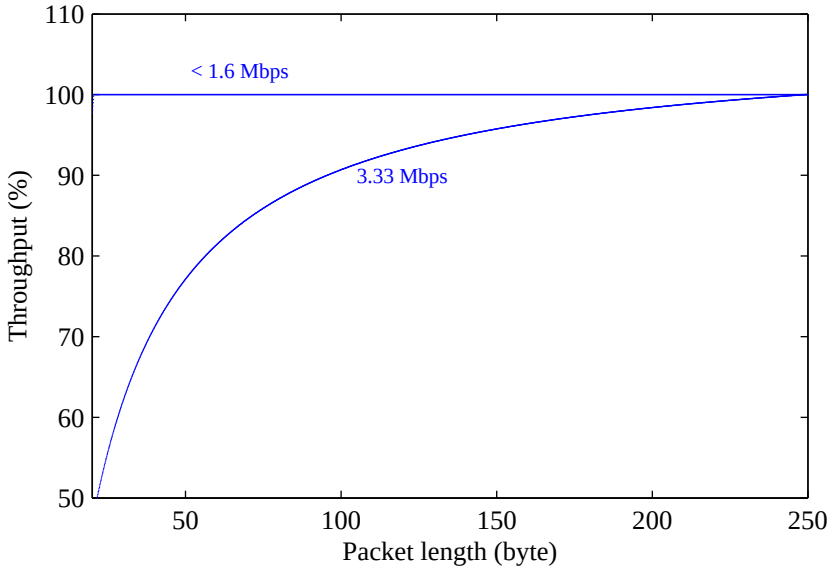


Figure 6.32: Throughput of network in terms of packet length with unoptimized time slot duration (using $h_{max} = 3$)

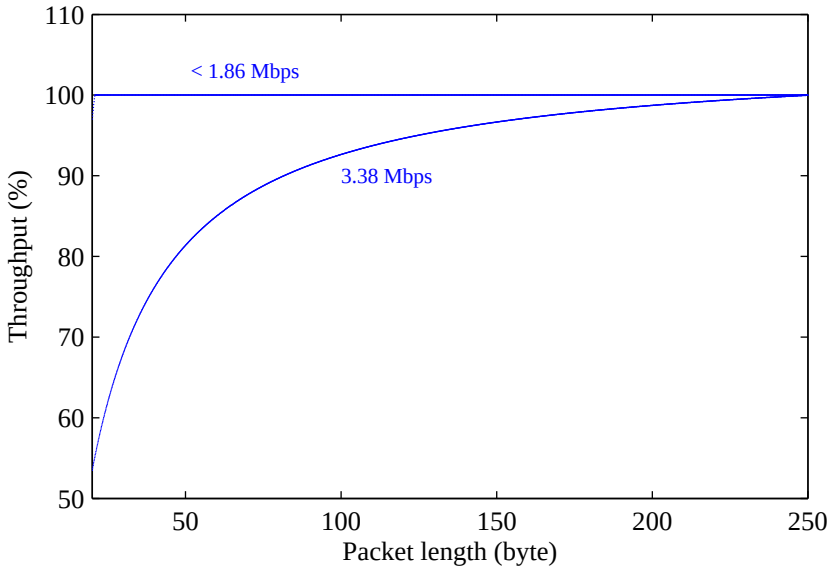


Figure 6.33: Throughput of network in terms of packet length; optimisal time slot duration.

CDR modules and a time synchronization circuit implemented in the MAC layer. The operation and analysis of the aforementioned parts is explained in detail in the Chapter 5.

Power consumption and speed of the circuit depend strongly on the utilized technology. A 0.35 µm technology was used to evaluate the functionality of the circuit, but since it is a fully digital circuit, the same design can be easily implemented in newer technologies to work at even higher speed and or to lower power consumption.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The main contribution of this dissertation is the design of a BAN system for measuring human locomotion parameters in the most practical, comfortable and non-invasive way. The carried out results and designed system can also be used in many BAN applications, especially systems that include a large number of sensors embedded in textile and high data-rate communication. The system consists of several hardware, software, and protocol modules and whole of the system has been designed from the scratch.

Chapter 3 introduced the SRMCF routing protocol. This is a reactive, energy-efficient routing protocol. It is established on the basis of SR concepts for ad hoc networks, and MCF methods for heterogeneous WSNs. The protocol analytically has compared with MCF protocol and their performance have been evaluated by simulation and experimental implementation on TelosB motes. SRMCF also has been used as routing protocol of the wearable sensors in both first and second prototypes.

The first prototype of the system has been presented in Chapter 4. It consists of several parts of a BAN including SNs, BS, CPM and connection between the SNs, which is established with conductive yarns and all end-to-end communication are based on packet switching. The functionality of each part of the system has been described and the behavior of them evaluated by both simulation and experimental results. SN consume low amount of power but buffering of packets in the intermediate nodes increases the end-to-end delay and also increases the power consumption when processing packets for routing to the final destination. The routing packets in microcontroller also increases the end-to-end delay due to packet reading via SPI port. The observed drawbacks of using packet switching in the first prototype has been taken into account and improved with the second prototype.

In Chapter 5, a very small, fully digital open-loop CDR method has been described and evaluated. The receiver clock comes from a local, autonomous oscillator that is running at a frequency of twice of the data rate. Synchronization is based on open-loop selection of the correct phase

of the clock in the receiver, synchronously with incoming signal. The achieved performance represents a trade-off between closed-loop and open-loop methods. The relatively higher jitter of the recovered clock is compensated by several favorable characteristics. The circuit size is much smaller than previously reported implementations, because it only consists of 8 logic elements and occupies 0.0022 mm^2 in the final IC. In that chapter, the time synchronization protocol for establishing time synchronization between sensor nodes of the wearable part of the system has been also introduced. The synchronization is based on one-way master-to-slave message exchange implemented in the MAC layer. Experimental evaluation with IC implementation shown an average one-hop clock skew of 4.6 ns that is the time required for signal propagation from sender output to the receiver input. Based on theoretical calculations, in a multi-hop network, the global average time skew grows linearly with hop count; this is supported by the experimental results. The sub-microsecond skew values provided by this approach satisfy the requirements of many BAN applications. The proposed circuit achieves the maximum synchronization performance that could be achieved by PTP, but with fewer timing messages and calculations, less complexity and better energy efficiency.

Chapter 6 introduced the second SN prototype. This prototype is based on hybrid circuit and packet switching implemented on an IC. Such a hybrid architecture with a TDM MAC protocol for scheduling the packet transmission provides less power consumption, less buffering in intermediate nodes and less end-to-end delay. The circuit also contains a high precision time synchronization part implemented in MAC layer, which was analyzed and evaluated in Chapter 5. For decreasing the packets serving time, the routing of data packets acquired by SNs to the BS is implemented inside the IC with the capability of direct accessing to the packets inside the buffer. The IC was fabricated in 0.35 µm CMOS technology and all the experimental results confirms significant improved behavior of the hybrid switching in comparison with only packet switching as used in the first prototype.

7.2 Future Work

The data rate obtained with the implemented circuit is 35 Mbps, sufficient for almost all BAN application. Nevertheless, it can be increased with some changes to the BS. All SNs and also the BS are equipped with one receiver. If the number of receivers in the BS is increased, then it can receive more packets simultaneously, leading to higher data rate. In this case, the BS architecture would need to be updated.

The system described in this dissertation includes a CPM, which works as an interface between the wearable platform and a computer. Although the obtained results confirm that the system is able to transmit the packets generated by sensors to a computer, combining BS and CPM in a single node will significantly improve the overall performance of the system. In fact, BS will be able to communicate directly with a computer. However, the size increase of the BS has to be taken into account.

Bluetooth, USB and MicroSD are the current interfaces between wearable system and computer. The reading and writing of these interfaces is performed by a microcontroller. If the network ASIC of BS is equipped with a circuit and extra ports (RS232 and SPI) to directly access to the aforementioned interfaces, then it will not be necessary to involve the microcontroller in that part of communication.

The switching part of the IC that performs circuit switching is a fully digital circuit. Another option is to use an analog switch in a mixed-signal IC, which may reduce switching time. It should be noted that it will be necessary to update the circuit switching protocol to meet the conditions in analog environment.

Supplying power to several sensors is one of the challenges in wearable systems. It goes without saying that using a single battery is much easier and comfortable than having several batteries for each sensor. The resistivity of conductive yarns is high, but using several yarns in parallel would make it possible to use them for power distribution. Another alternative solution is using PLC techniques, which would allow the use of single wires for both power supply and communication.

Miniaturization is a key factor in BAN applications. Consider a System on Chip (SoC) that integrates network module, processing module (CPU) and also the circuits necessary for sensing, such as ADC, in a single chip. Such an IC would make the sensor significantly smaller (in the range of a few square millimeters). In addition, using a smaller technology, such as 130 nm, 65 nm or even smaller, will reduce both circuit size and power consumption.

7.3 Publications

The work done for this dissertation has resulted in the following publications (proceedings of international conferences and journals):

1. F. Derogarian, J. C. Ferreira, and V. M. G. Tavares, “A routing protocol for WSN based on the implemention of source routing for minumum cost forwarding method,” in Fifth Intl. Conf. Sensor Tech. Appl. SENSORCOMM2011, Aug. 2011.

2. A. Zambrano, F. Derogarian, R. Dias, M. J. Abreu, A. Catarino, A. M. Rocha, J. M. da Silva, J. C. Ferreira, V. M. G. Tavares, and M. V. Correia, “A wearable sensor network for human locomotion data capture,” in 9th Intl. Conf. Wearable Micro and Nano Technologies for Personalized Health, Phealth12, Jun. 2012.

3. F. Derogarian, J. C. Ferreira, and V. M. G. Tavares, “ Using a wired body area network for locomotion data acquisition,” in 27th Conf. on Design of Circuits and Integrated Systems, DCIS2012, November 2012.

4. F. Derogarian, J. C. Ferreira, and V. M. G. Tavares, “ Design and implementation of a circuit for mesh networks with application in body area networks,” in Fifth Euromicro Conf. on Digital System Design, DSD2012, September 2012.

5. A. Catarino, A. M. Rocha, M. J. Abreu, F. Derogarian, J. da Silva, J. C. Ferreira, V. M. G. Tavares, M. Correia, and R. Dias, “E-Legging for Monitoring the Human Locomotion Patterns,” Journal of Textile Engineering, vol. 59, no. 6, pp. 153–158, Dec. 2013.

6. F. Derogarian, J. C. Ferreira, and V. M. G. Tavares, “ A time synchronization circuit with an average 4.6 ns one-hop skew for wired wearable networks,” in 17th Euromicro Conf. on Digital Systems Design, DSD2014, Aug. 2014.

7. F. Derogarian, J. C. Ferreira, and V. M. G. Tavares, “ Design and implementation of hybrid circuit/packet switching for wearable systems,” in 23rd Intl. Symposium on Industrial Electronics, ISIE2014, Jun. 2014.

Paper 7 resulted in the invitation to submit an extended version for a special issue of Microprocessors and Microsystems: Embedded Hardware Design. The paper has been accepted for publication.

Two other papers have been submitted to international journals:

1. F. Derogarian, J. C. Ferreira, and V. M. G. Tavares, “A Small Fully Digital Open-loop Clock and Data Recovery Circuit for Wired BANs”, accepted for publication, Intl. Journal of Circuit Theory and Applications.

2. F. Derogarian, J. C. Ferreira, and V. M. G. Tavares, “Analysis and Evaluation of an Energy-Efficient Routing Protocol for WSNs”, submitted to Springer Intl. Journal of Wireless Networks.

3. F. Derogarian, J. C. Ferreira, and V. M. G. Tavares, “A Precise and Hardware-Efficient Time Synchronization Method for Wearable Wired Networks”, submitted to IEEE Sensors Journal.

Appendix A

Schematics for Sensor Node, Base Station and Central Processing Module

Chapters 4 and 6 describe the first and second prototypes of the different BAN elements. Here their schematics are presented with more details about the components and connections. The circuits were designed with OrCAD, a high performance circuit and PCB layout design tool.

Figure A.1 shows the schematic of a SN and the BS. From a hardware perspective, both SNs and BS are similar. So, the same circuit can be set to work as SN or BS. In fact, it is the software running on microcontroller U1 that determines whether the node works either as SN or BS.

The low-power, 16-bit microcontroller (PIC24F64GA104) in the center of the schematic is connected to the network component and also to the EMG sensor, accelerometer and gyroscope. The connections of U1 to the peripheral devices are as follows:

1. The physical and MAC layers of the network are implemented on an FPGA, which is connected to U1 through an SPI port.

2. The EMG module is a circuit located on a separate board and connected to the main board via the J5 connector. This module was designed by Ruben Dias for the ProLimb project. The output of EMG module is analog. U1 contains a 10-bit and up to 13 channel analog-to-digital converter; one of them is used to convert the EMG signals to digital.

3. Both accelerometer and gyroscope are also on separate modules equipped with an I2C port. J3 connects these sensors to the main board and microcontroller as well.

4. Connector J4 provides 3 types of connection: first, it is used to program the microcontroller; second, it serves as a SPI port to connect to the CPM when the board works as a BS; third, it can be used as a RS232 port for debugging purpose.

U2 is a low-power FPGA from Actel (AGLN125). The physical and MAC layers are implemented on this FPGA. It contains 3,072 cells (2509 used in this work) and an 36 kbit RAM, which is used as a buffer for the packets. This FPGA is programmable via J7 and is connected to the Port module by J6.

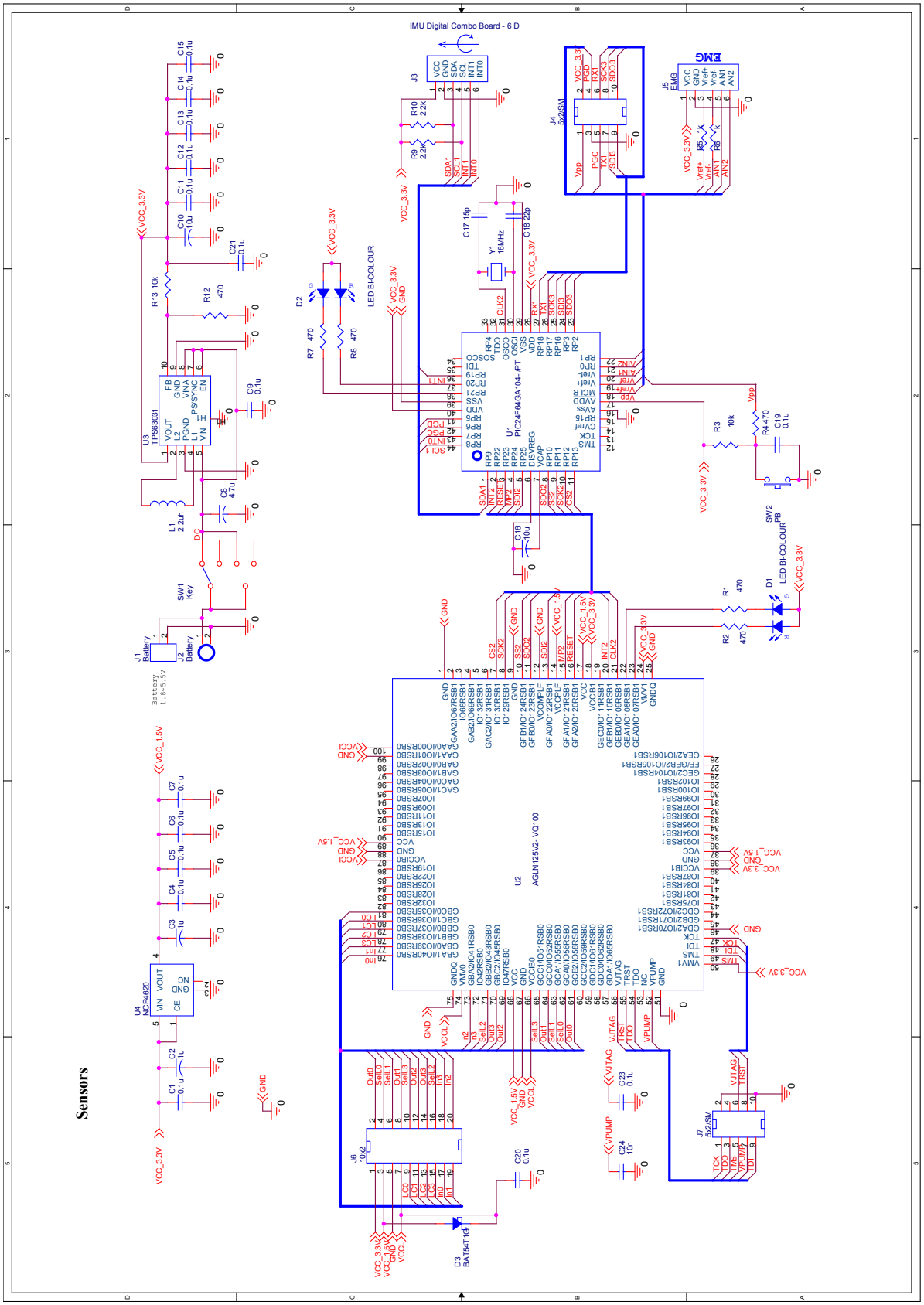


Figure A.1: Schematic of SN and BS.

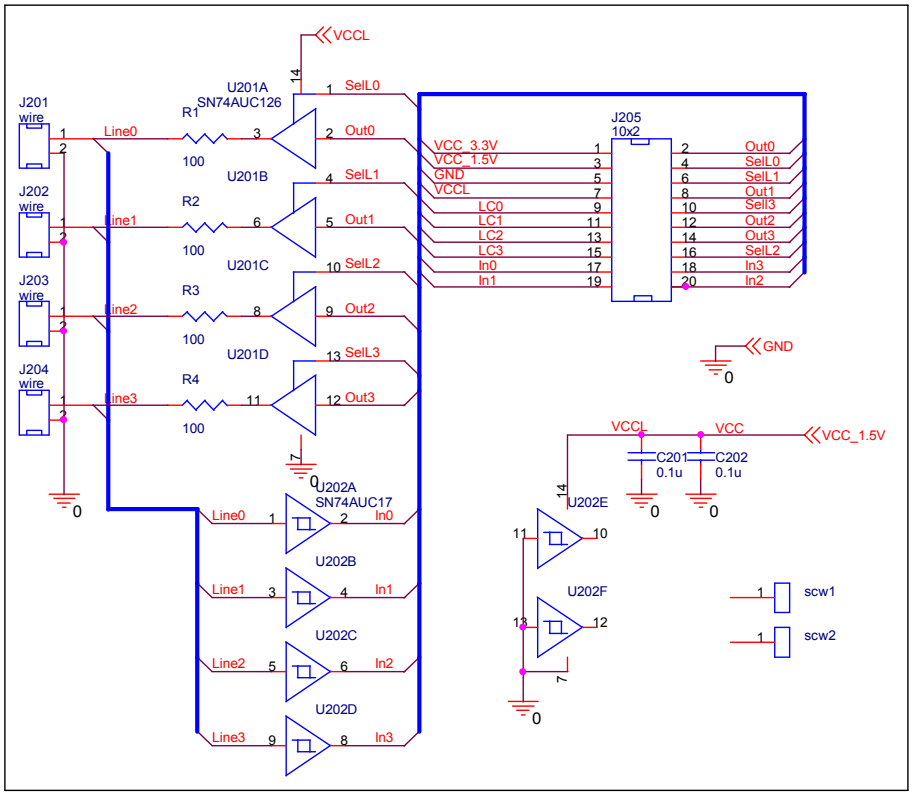


Figure A.2: Schematic of the port module.

The power supply is a single coin type battery (LIR2450) connected to J1. Except for the FPGA core, all other devices require a stable 3.3 V supply to work properly. The nominal battery voltage is 3.6 V and it goes down to 2.75 V during use. So a buck-boost dc-to-dc converter is needed to generate 3.3 V at the output. For that, the TPS63031, a high-efficiency single-inductor buck-boost converter from Texas Instruments, is used. NCP4620, a linear regulator from ON Semiconductor, is used to generate 1.5 V for FPGA core and for line drivers.

Figure A.2 shows the port module. This module directly mounts on the main board. The SN74AUC126 contains 4 independent line drivers with 3-state outputs and is used to send data. Each output is disabled when the associated output-enable (OE) input is low. SN74AUC17 is a 4 schmitt trigger and is used to receive data and generate clear digital signals at the output. The *SelLx* lines control whether the ports act as sender or receiver.

The CPM circuit is shown in Fig. A.3. This circuit uses the same PIC24F64GA104 microcontroller as is used in the SN prototype. The connector J106, like J4 in SN circuit, provides 3 types of connection: first, it is used to program the microcontroller; second, as an SPI port to connect to the CPM when the board works as a BS; third, as a RS232 port for debugging purpose. Connector J104 provides a USB connection via a FTDI USB module. The output of this module is compatible with RS232 and the microcontroller can send or receive data directly from a computer.

The Bluetooth module WT12 also connected by a second RS232 port to the microcontroller and is used for wireless connection. The SDCard memory is compatible with the SPI port for fast

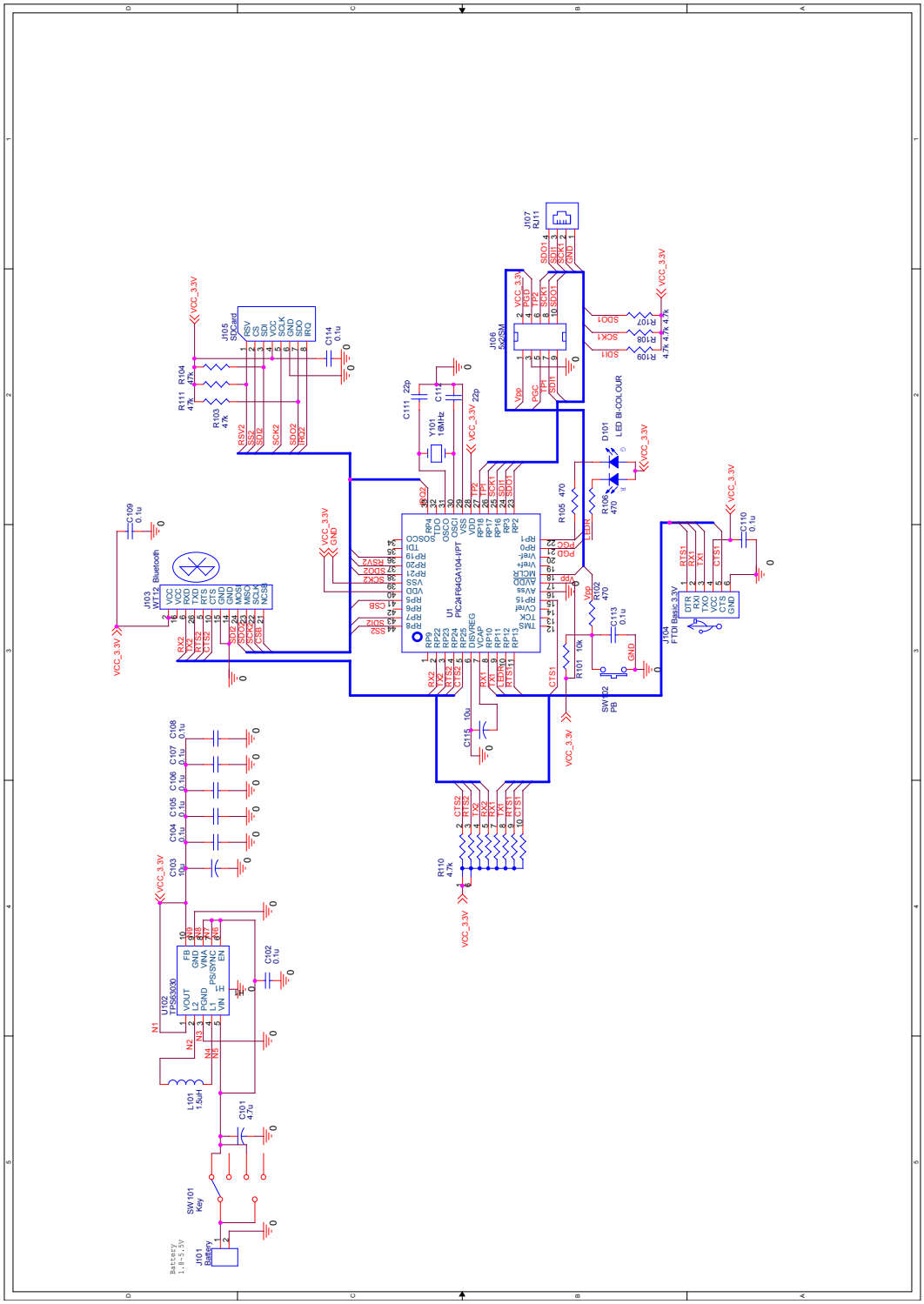


Figure A.3: The CPM schematic.

data read/write operation. For that, the second SPI port of microcontroller is used.

Figure A.4 shows the second prototype of the SN and BS. This circuit is used to evaluate the designed ASIC U2 described in Chapter 6. The voltage regulator and microcontroller are the same as used in the first prototype. U4 and U5 are highly accurate current sensors that are used to measure the current consumption of the circuit.

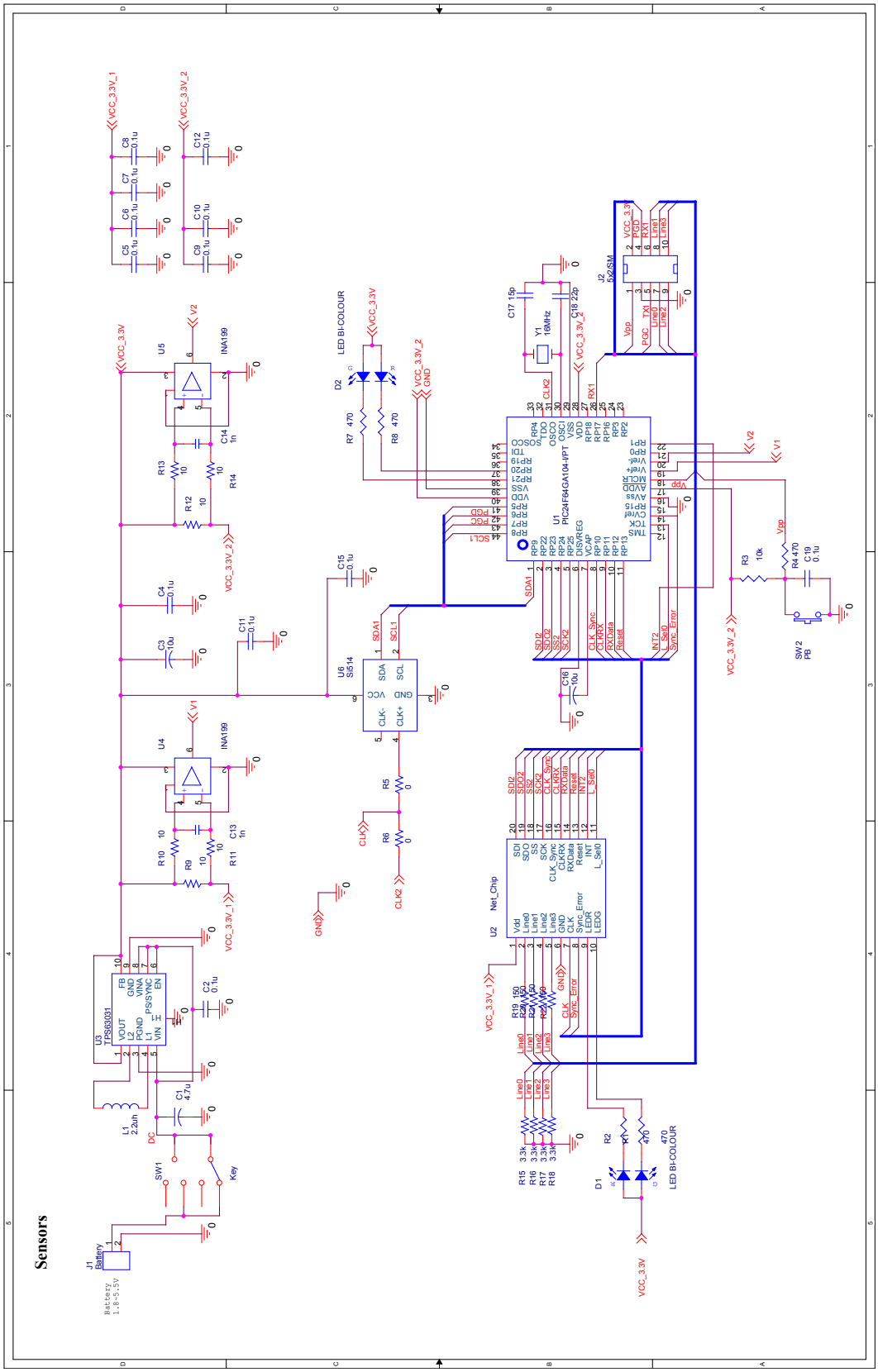


Figure A.4: Schematic of the second sensor node prototype.

Appendix B

ASIC Testbench Examples

Section 6.3 mentions that some testbenches have been designed for validation of the IC in a network. This appendix illustrates with some example how the testbenches can be used to simulate networks of sensors. A few tasks are considered: for each of them the simulation results are shown and explained. All simulation results were obtained after placement and routing of the chip with Cadence SocEncounter. The final chip was simulated with the NCSim simulator through NClaunch. Finally, the SimVision waveform viewer was used for plotting the signals.

A testbench consisting of BS and 4 SNs is portrayed in Fig. B.1. Node SN2 is the *near-node* for both SN3 and SN4 and its own *near-node* is SN1. The clock frequency of the circuits is 20 MHz with ± 2 % tolerance. The clock tolerance is intentionally applied to evaluate the IC performance in spite of the inaccuracy of crystal quartz oscillators, which is normally around ± 0.003 %. The SPI communication structure is presented in Section 4.3.2.2. Before starting normal operation, each SN has to be configured. Figure B.2 shows the configuration process for SN4. The process is as follows:

1. First, the data rate is set to half the system clock frequency (0x80).
2. The line coding is set to 8b9b (0x00). This value can vary from 0x00 (for 8b9b coding) to 0x07 (for 64b65 coding).
3. For each SN, the port connected to its *near-node* has to be set. For SN4, it is port 1 (0x01) .
4. Each node has an network ID, which can be any 8-bit number except zero (reserved for the BS). Here, the ID of SN4 is 0x12.
5. All the interrupts generated by internal modules can be enabled or disabled. For SN4, the two enabled interrupts are: reception and error on either reception or transmission.
6. The time synchronization module has some registers for configuring its operation. The first register is used to enable, disable or request *TC* and *TS* synchronization and the other registers are used for the *TC* value and offset. The time synchronization module of SN4 is enabled and the values of *TC* and offset are 0x0101 and 0x0000, respectively.

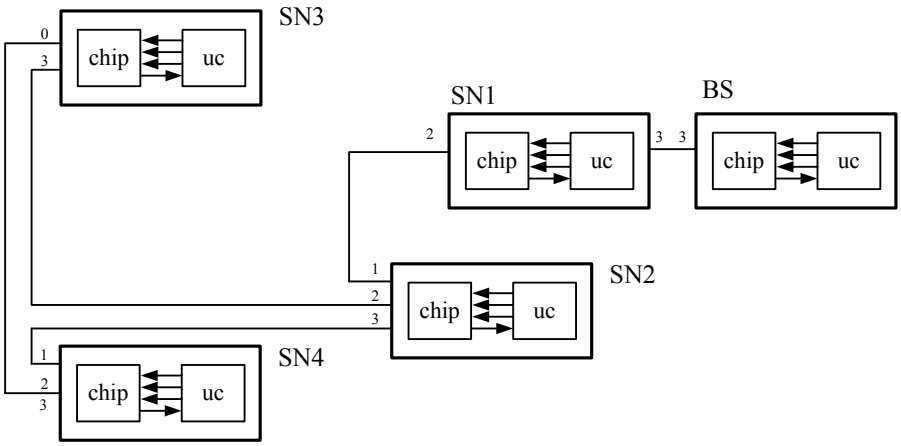


Figure B.1: A testbench with a BS and four sensor nodes.

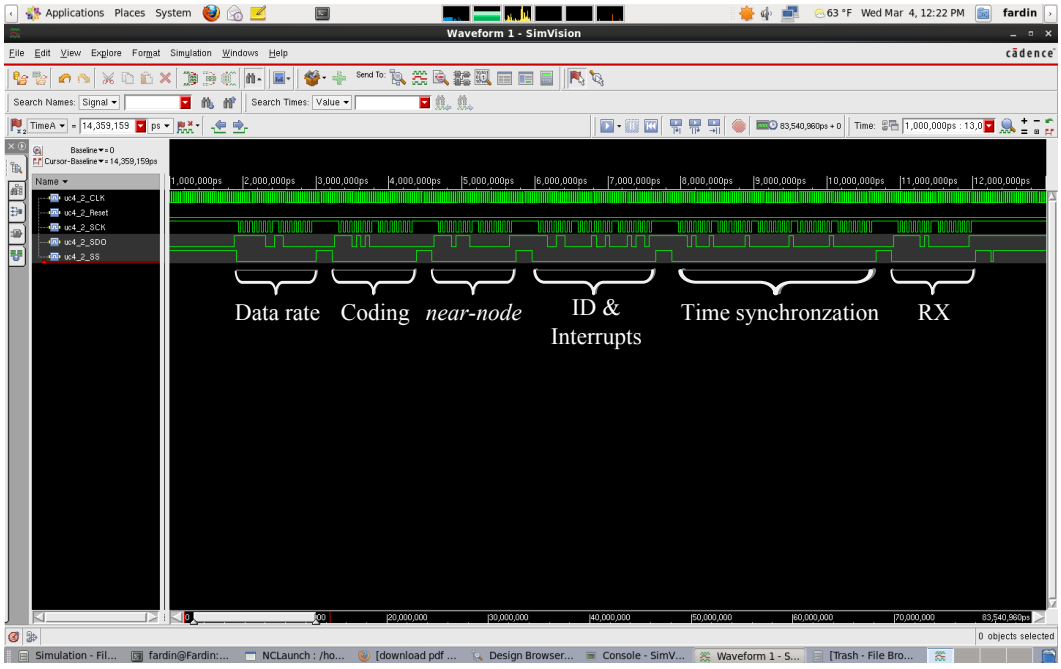


Figure B.2: Configuring the communications ASIC.

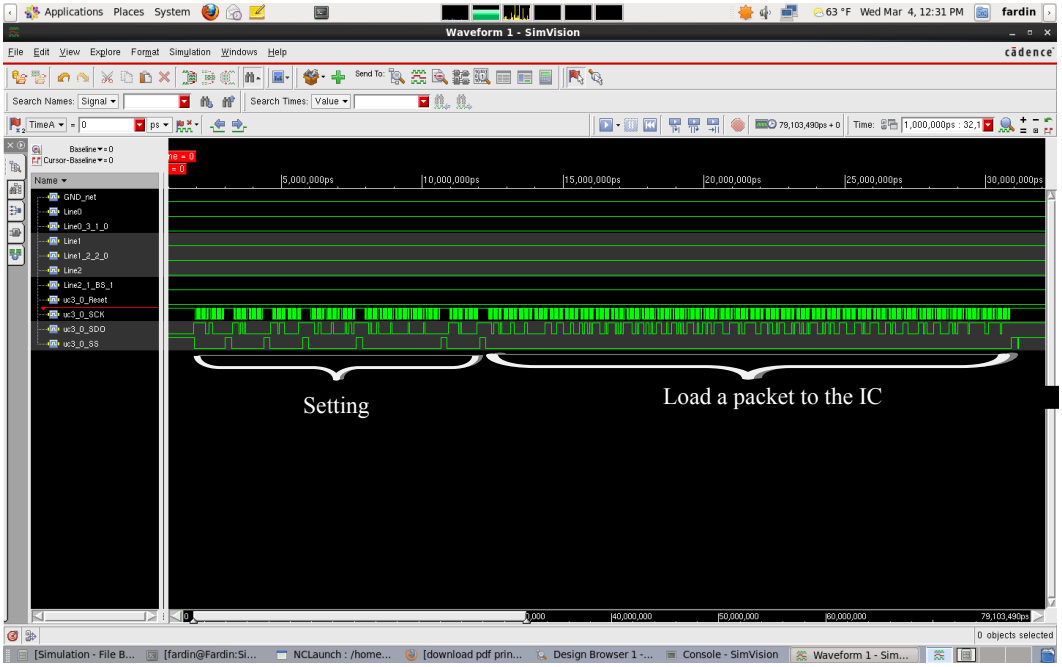


Figure B.3: Loading a packet to a buffer in the ASIC.

7. The configuration register of the *RX* mode determines the receiver and clock recovery operation modes. Here, the clock recovered by *RX* is set to be synchronized with both falling and rising edges of incoming NRZI encoded data stream.

After configuration, the ASIC is ready to play its role in the network. A packet transmission starts whenever the microcontroller loads data to the ASIC via the SPI port. Figure B.3 illustrates the loading of a data packet immediately after the configuration process. At the end of loading (SS pin changes to high level), ASIC recognizes the existence of the packet in the buffer and starts to process the packet as explained in Section 6.2.1. The microcontroller determines the destination node and switching mode (in case of transmission to the BS) whenever it loads a packet to ASIC. Figure B.4 shows the packet delivery to the BS suing circuit switching. The simulation results are very similar to the experimental results presented in Section 6.4 (obtained with an oscilloscope).

Figure B.5 shows the time synchronization process performed by SN2. Initially, *Clk-sync* is out of synchronization. The microcontroller of SN2 sends a command to the ASIC requesting *TC* synchronization. Then SN2 sends a timing message *Request* to SN1 and synchronizes its own *TC* counter. The result of synchronization appears on *Clk-sync* clock. After that, the node sends another message to synchronize its *TS* counter. Meanwhile, SN2 replies to *Request* messages from SN3 and SN4. This process also has been observed experimentally to match exactly the one simulated here.

The simulation results presented here are just some of the possible examples of using the testbench infrastructure developed for this work. These testbenchs can be modified for any number of nodes and any kind of network and the results will be very near to the actual implementation.

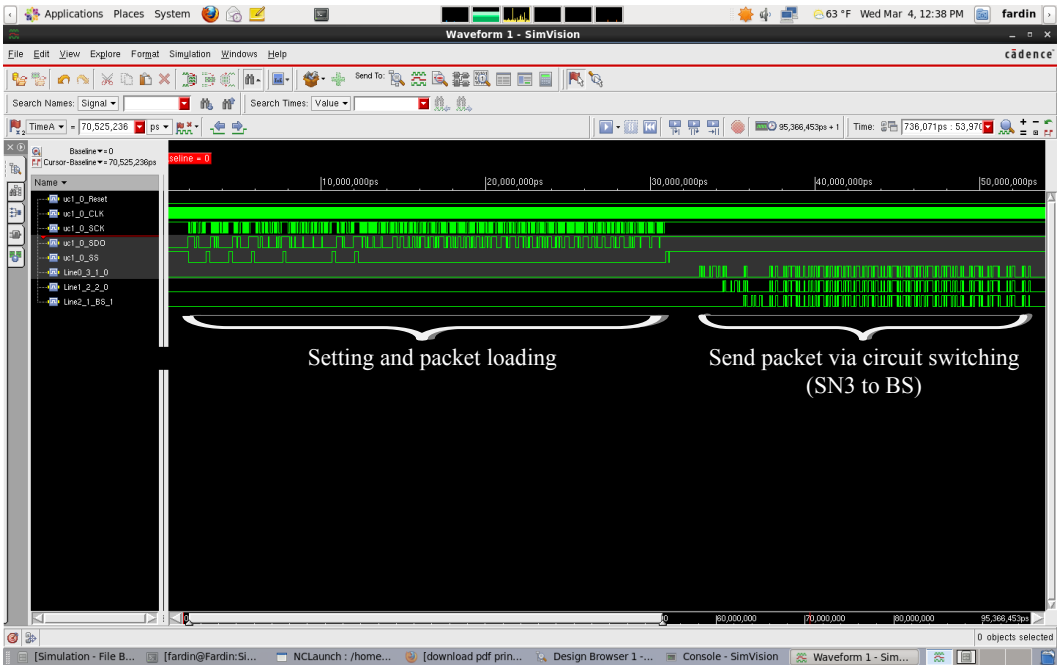


Figure B.4: Sending a packet by circuit switching immediately after loading it to the buffer.

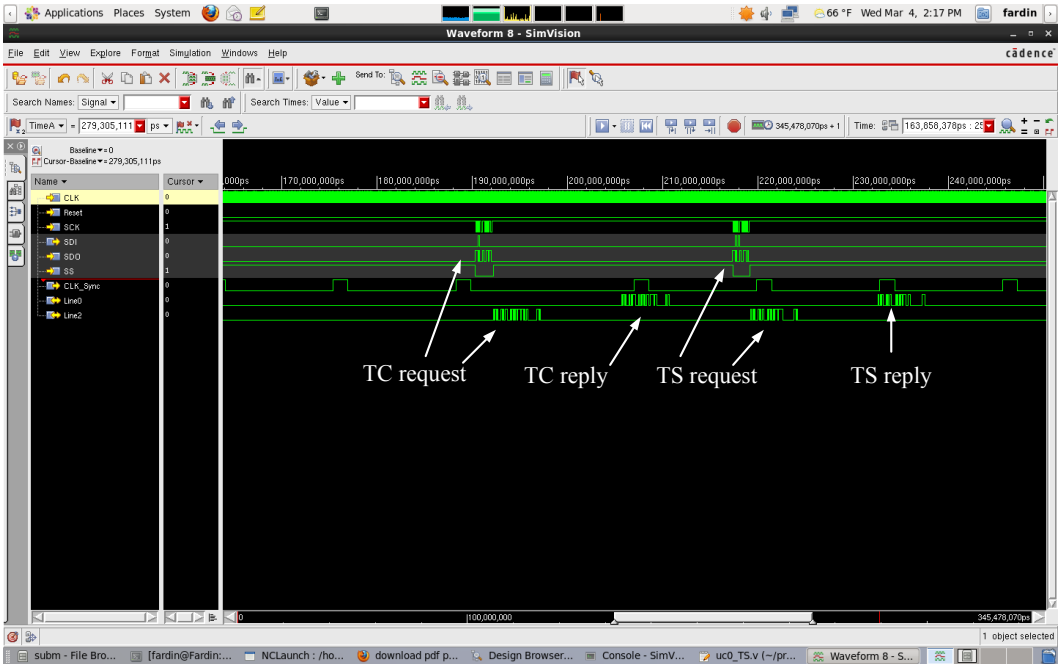


Figure B.5: Simulation of TC and TS synchronization.

Appendix C

Acquisition of Locomotion Data

At the end of Chapter 4, some of the results obtained with the sensors used to measure lower limb activity have been shown. I In this appendix, more results obtained with a system including 8 sensors are presented. The results were obtained in the context of the ProLimb project. The program for display was developed by Ruben Dias. The BAN designed for this dissertation was used to handle all data transfers.

A flexible wearable textile produced by André Catarino (University of Minho) for gait analysis is shown in Fig. C.1. It uses conductive yarns with R=1.5 Ω/cm and L=8 nH/cm. The woven conductive yarns can be clearly seen on the garment. sEMG electrodes capture signals properly if a good reference point on the body is available. Usually the reference point is a part of body with very weak sEMG signals, e.g. elbow or knee. Hence, the big point illustrated in Fig. C.1 on the knee is the reference point. Figure C.2 depicts the interconnections of the aforementioned garment. The same setup is also used for the e-legging in Fig. C.3.

All sensors are equipped with sEMG electrodes but only SN1, SN3, SN5 and SN7 have accelerometer and gyroscope. The sEMG signal acquisition module was designed by Ruben Dias for the ProLimb project. In fact, for each right and left leg and thigh, one accelerometer and one gyroscope is enough to capture kinematic activity. In the gait exercise, 18 steps were performed in a straight line. The sampling rate of the accelerometer and gyroscope is 50 Hz with 16-bit resolution in 3 dimensions; the sEMG signals are sampled at 1 kHz with 16-bit resolution. Figures C.4 to C.7 illustrate the sEMG, acceleration and angular rate signals captured with SN1, SN3, SN5 and SN7, respectively, during the gait exercise. The real-time signals have been captured via a Bluetooth connection while the subject was walking. Figures C.8 to C.15 also depict the sEMG signals for 18 steps. It can be seen that the wearable platform is able to acquire and transmit a sequence of signals which are timely synchronized and clearly distinguishable, and that can be used for characterizing muscle activity.

Figure C.1: E-legging for capturing human locomotion [from ProLimb project proposal].

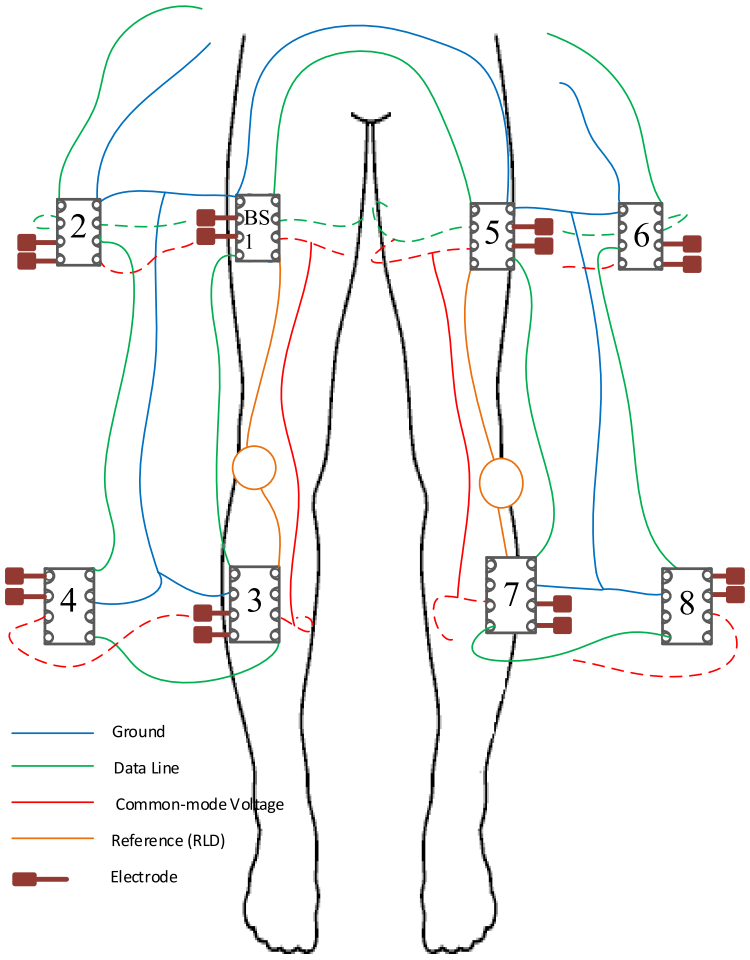


Figure C.2: Interconnection diagram of wearable platform.

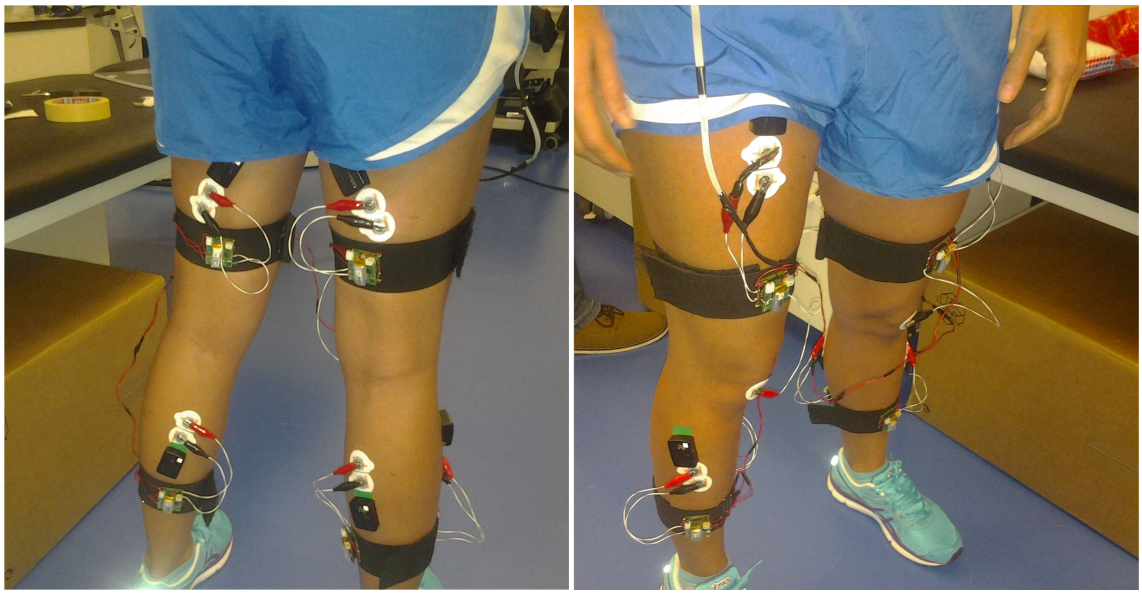


Figure C.3: Interconnection diagram of e-legging.

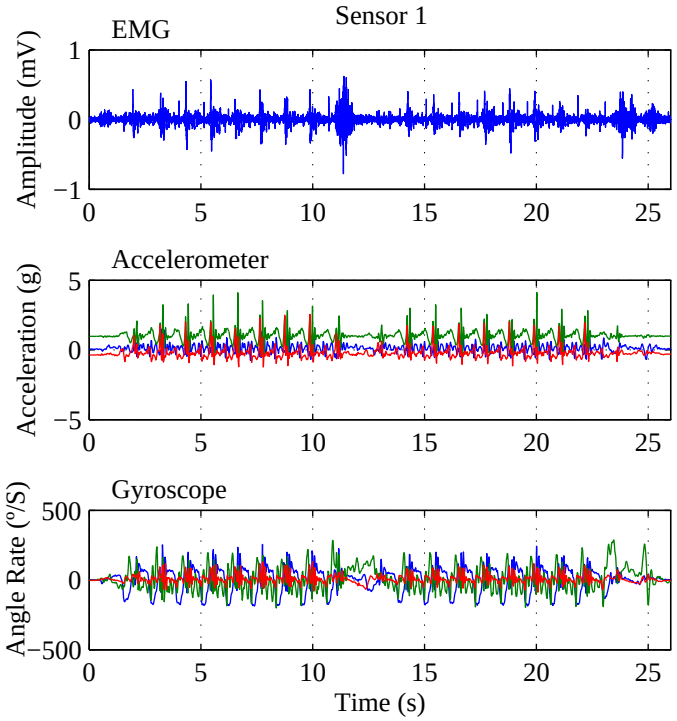


Figure C.4: sEMG, acceleration and angular rate signals captured with the SN1.

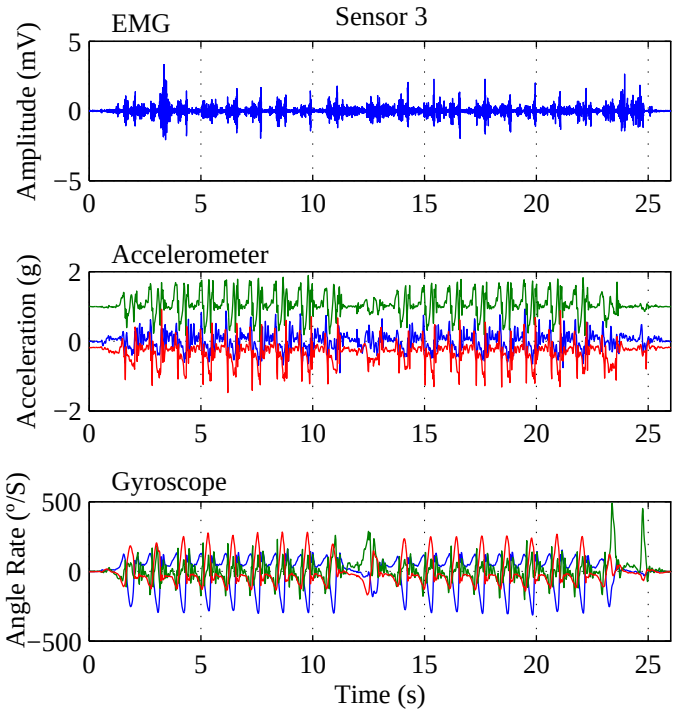


Figure C.5: sEMG, acceleration and angular rate signals captured with the SN3.

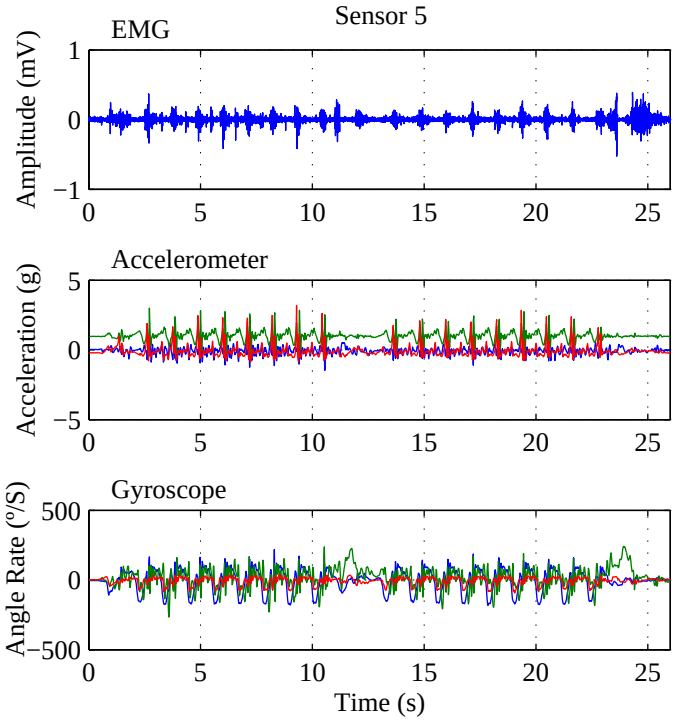


Figure C.6: sEMG, acceleration and angular rate signals captured with the SN5.

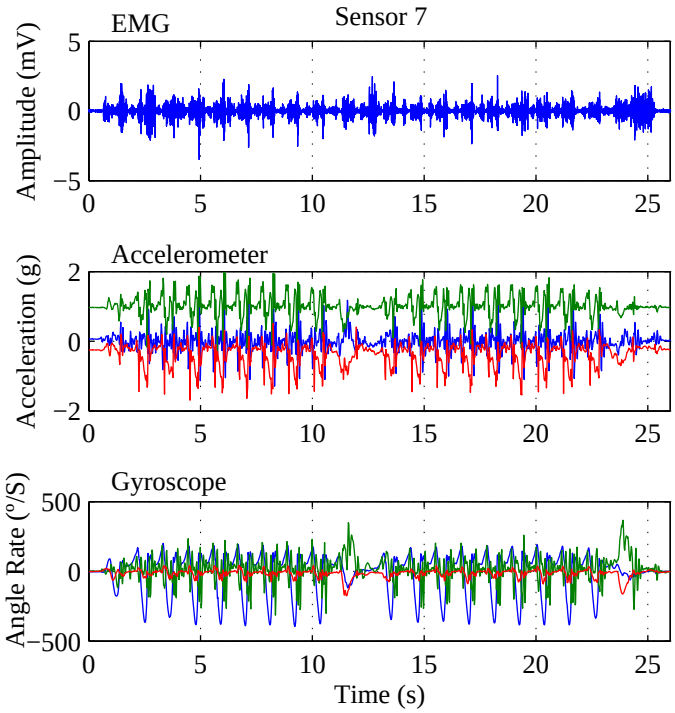
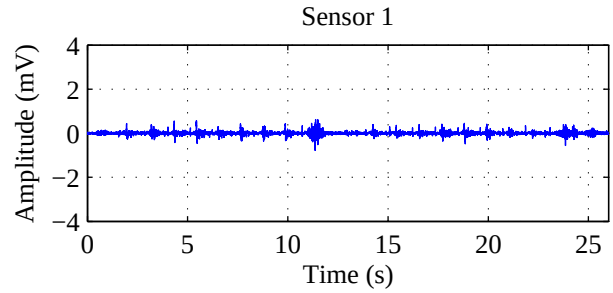


Figure C.7: sEMG, acceleration and angular rate signals captured with the SN7.



z

Figure C.8: sEMG signals captured with SN1.

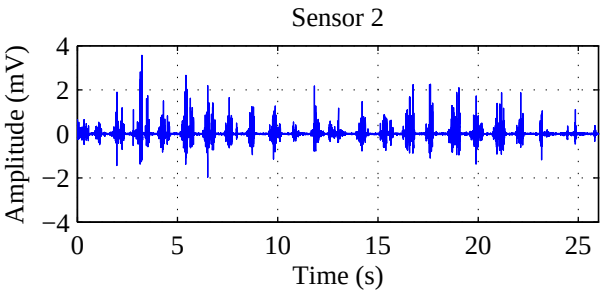


Figure C.9: sEMG signals captured with SN2.

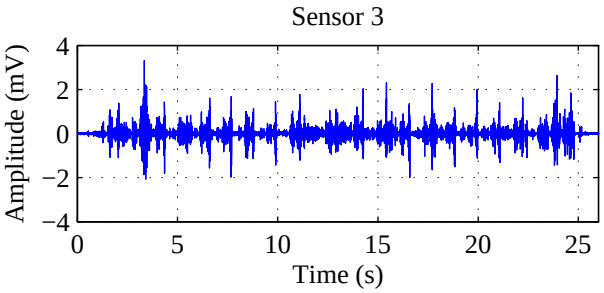


Figure C.10: sEMG signals captured with SN3.

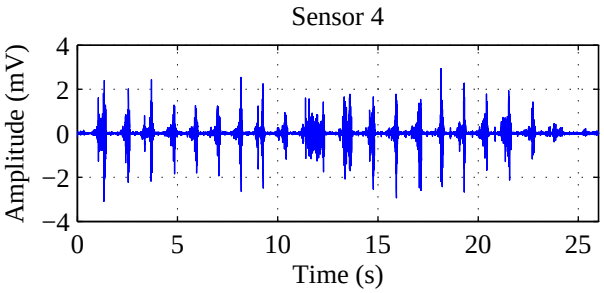


Figure C.11: sEMG signals captured with SN4.

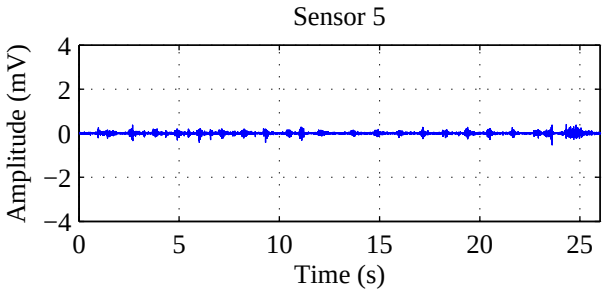


Figure C.12: sEMG signals captured with SN5.

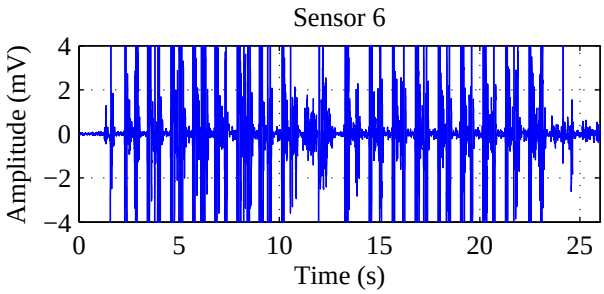


Figure C.13: sEMG signals captured with the SN6.

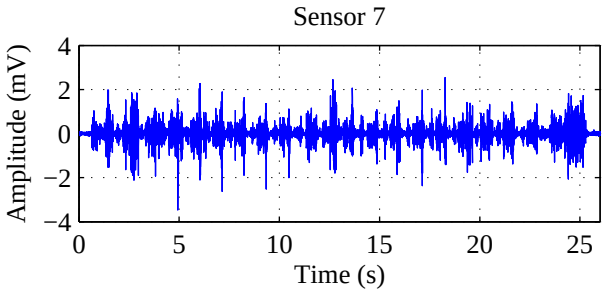


Figure C.14: sEMG signals captured with the SN7.

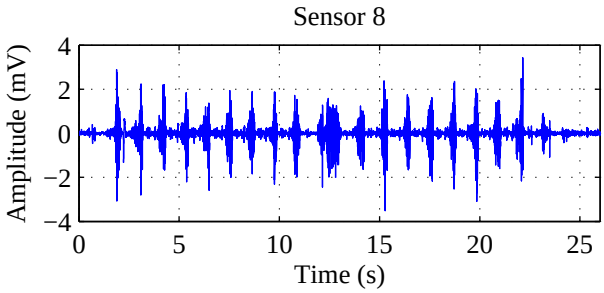


Figure C.15: sEMG signals captured with the SN8.

References

[AAWA08] S. Ahmad, I. Awan, A. Waqqas, and B. Ahmad. Performance analysis of DSR and extended DSR protocols. In *2nd Asia Intl. Conf. on Modeling Simulation*, pages 191–196, May 2008.

[AC14] A.R. Ansari and Sunghyun Cho. Human body: The future communication channel for WBAN. In *18th IEEE Intl. Symp. on Consumer Electronics (ISCE)*, pages 1–3, June 2014.

[AER08] L.K. Alazzawi, A.M. Elkateeb, and A. Ramesh. Scalability analysis for wireless sensor networks routing protocols. In *22nd Intl. Conf. on Advanced Information Networking and Applications, Workshops (AINAW)*, pages 139 –144, march 2008.

[AME06] Koubaa A., Alves M., and Tovar E. A comprehensive simulation study of slotted csma/ca for IEEE 802.15.4 wireless sensor networks. In *IEEE Intl. Workshop on Factory Communication Systems*, pages 183–192, 2006.

[AMM08] M. Asim, H. Mokhtar, and M. Merabti. A fault management architecture for wireless sensor network. In *Intl. Conf. on Wireless Communications and Mobile Computing (IWCMC08)*, pages 779–785, 2008.

[ANI11] Z.M. Ashari, A.N. Nordin, and M.I. Ibrahimy. Design of a 5 GHz phase-locked loop. In *IEEE Regional Symp. on Micro and Nanoelectronics (RSM)*, pages 167–171, September 2011.

[ANM+12] Rahim A., Javaid N., Aslam M., Rahman Z., Qasim U., and Khan Z.A. A comprehensive survey of MAC protocols for wireless body area networks. In *17th Intl. Conf. on Broadband, Wireless Computing, Communication and Applications (BWCCA)*, pages 434–439, November 2012.

[ASS+08] Junichi Akita, Toru Shinmura, Shigeru Sakurazawa, Keisuke Yanagihara, Mihoko Kunita, Masashi Toda, and Kunio Iwata. Wearable electromyography measurement system using cable-free network system on conductive fabric. *J. of Artificial Intelligence in Medicine*, 42(2):99–108, February 2008.

[ASSC02] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *IEEE Magazine on Communications*, 40:102–114, August 2002.

[Atm11] Atmel. *8-bit Atmel Microcontroller with 128 KBytes In-System Programmable Flash*, April 2011. Rev. 2467X–AVR–06/11.

[AY05] Kemal Akkaya and Mohamed Younis. A survey on routing protocols for wireless sensor networks. *J. of Ad Hoc Networks*, 3:325–349, 2005.

[AYB13] A.A. Abbasi, M.F. Younis, and U.A. Baroudi. Recovering from a node failure in wireless sensor-actor networks with minimal topology changes. *IEEE J. of Vehicular Technology*, 62(1):256–271, Jan 2013.

[BB10] B.Baranidharan and B.Shanthi. Article:a survey on energy efficient protocols for wireless sensor networks. *Intl. J. of Computer Applications*, 11(10):35–40, December 2010. Published By Foundation of Computer Science.

[BBA14] Shahina Begum, Shaibal Barua, and Mobyen Ahmed. Physiological sensor signals classification for healthcare using sensor data fusion and case-based reasoning. *J. of Sensor*, 14(7):11770–11785, July 2014.

[BBI+07] Latre B., Braem B., Moerman I., Blondia C., Reusens E., Joseph W., and Demeester P. A low-delay protocol for multihop wireless body area networks. In *Mobile and Ubiquitous Systems: Networking Services, 2007. MobiQuitous 2007. Fourth Annual International Conference on*, pages 1–8, Aug 2007.

[BBI+11] Latré Benoît, Braem Bart, Moerman Ingrid, Blondia Chris, and Demeester Piet. A survey on wireless body area networks. *J. of Wireless Networks*, 17(1):1–18, January 2011.

[BCK11] F.Z. Benhamida, Y. Challal, and M. Koudil. Efficient adaptive failure detection for query/response based wireless sensor networks. In *Wireless Days (WD) (IFIP)*, pages 1–6, oct. 2011.

[BDMS13] L. Barletta, R. Disaro, M. Magarini, and A. Spalvieri. Post-filter optimization in timing recovery based on square-law detection. *IEEE J. of Photonics Technology Letters*, 25(9):821–824, 2013.

[BGdMT00] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi. *Queueing Networks and Markov Chains*. John Wiley & Sons, Inc, 2nd edition, 2000.

[BMIM07] Costas Busch, Malik Magdon-Ismail, and Marios Mavronicolas. Universal bufferless packet switching. *SIAM J. of Computing*, 37(4):1139–1162, November 2007.

[BNIA09] S. Bushnaq, T. Nakura, M. Ikeda, and K. Asada. All digital baseband 50Mbps data recovery using 5x oversampling with 0.9 data unit interval clock jitter tolerance. In *12th Intl. Symp. on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, pages 206–209, April 2009.

[Bre02] Stefano Bregni. *Synchronization of Digital Telecommunications Networks*. John Wiley and Sons, Ltd, 2002.

[BSM+12] A. Boulis, D. Smith, D. Miniutti, L. Libman, and Y. Tselishchev. Challenges in body area networks for healthcare: the MAC. *IEEE Communications Magazine*, 50(5):100–106, May 2012.

[CBS+11] Cristina Cano, Boris Bellalta, Anna Sfairopoulou, Miquel Oliver, and Jaume Barceló. Taking advantage of overhearing in low power listening WSNs: A performance analysis of the LWT-MAC protocol. *MONET*, 16(5):613–628, 2011.

[CcDgLs+10] Zhang Cheng-chang, Yan Dan-gui, Yang Li-sheng, Qi huai long, and Li Changyong. DLL-based multi-FPGA systems clock synchronization. In *5th IEEE Conf. on Industrial Electronics and Applications (ICIEA)*, pages 1420–1423, June 2010.

[CCR02] A. Bruce Carlson, Paul B. Crilly, and Janet C. Rutledge. *Communication Systems*. McGraw-Hill, 4th edition, 2002.

[CGV+11] Min Chen, Sergio Gonzalez, Athanasios Vasilakos, Huasong Cao, and Victor C. Leung. Body area networks: A survey. *J. of Mobile Networks and Applications*, 16(2):171–193, April 2011.

[Che03] Wai Kai Chen. *The circuits and filters handbook*. CRC Press, 2nd edition, 2003.

[CLCC09] Huasong Cao, V. Leung, C. Chow, and H. Chan. Enabling technologies for wireless body area networks: A survey and outlook. *IEEE Communications Magazine*, 47(12):84–93, Dec 2009.

[Con12] Contiki OS. .*The Open Source OS for the Internet of Things*, September 2012.

[Cro11] Crossbow Technology. *Micaz: 2.4Ghz Mote*, June 2011.

[Cro13] Crossbow Technology. *TELOSB*, February 2013.

[CW09] Fan-Ta Chen and Jen-Ming Wu. An extended phase detector 2.56/3.2 Gb/s clock and data recovery design with digitally assisted lock detector. In *IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, pages 1831–1834, May 2009.

[CWJ11] Chih-Lin Chen, Chua-Chin Wang, and Chun-Ying Juan. A fast-locking clock and data recovery circuit with a lock detector loop. In *13th Intl. Symp. on Integrated Circuits (ISIC)*, pages 332–335, December 2011.

[CYBY09] Namjun Cho, Long Yan, Joonsung Bae, and Hoi-Jun Yoo. A 60 kb/s to 10 mb/s adaptive frequency hopping transceiver for interference-resilient body channel communication. *IEEE J. of Solid-State Circuits*, 44(3):708–717, March 2009.

[CZ07] Zhiming Chen and Yueping Zhang. A modified synchronization scheme for impulse-based UWB. In *6th Intl. Conf. on Information, Communications Signal Processing*, pages 1–5, December 2007.

[DFT10] Fardin Derogarian, João C. Ferreira, and Vítor M. Grade Tavares. A routing protocol for WSN based on the implemention of source routing for minumum cost forwarding method. In Jens Martin Hovem, Joshua Ellul, Laurent Gomez, and Yenumula Reddy, editors, *5th Intl. Conf. Sensor Technologies and Applications (SENSORCOMM)*, pages 85–90. ThinkMind, August 2010.

[DFT14a] Fardin Derogarian, João C. Ferreira, and Vítor M. Grade Tavars. Design and implementation of hybrid circuit/packet switching for wearable systems. In *23rd Intl. Symp. on Industrial Electronics (ISIE)*, June 2014.

[DFT14b] Fardin Derogarian, João C. Ferreira, and Vítor M. Grade Tavars. A time synchronization circuit with an average 4.6 ns one-hop skew for wired wearable networks. In *17th Euromicro Conf. on Digital Systems Design (DSD)*, August 2014.

[DGV04] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *29th Annual IEEE Intl. Conf. on Local Computer Networks*, pages 455–462, Nov 2004.

[DP10] Waltenegus Dargie and Christian Poellabauer. *Fundamentals of Wireless Sensor Networks: Theory and Practice*. Wiley, 1st edition, 2010.

[DP14] K.G. Dangi and S.P. Panda. Challenges in wireless body area network-a survey. In *Intl. Conf. on Optimization, Reliabilty, and Information Technology (ICROIT)*, pages 204–207, Feb 2014.

[DPZ04] Richard Draves, Jitendra Padhye, and Brian Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *10th ACM Intl. Conf. on Mobile Computing and Networking*, MobiCom '04, pages 114–128, 2004.

[DR03] Michael Duck and Richard Read. *Data Communications and Computer Networks*. Prentice Hall, 2nd edition, 2003.

[EGE02] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. In *5th Symp. on Operating systems design and implementation (SIGOPS)*, pages 147–163, 2002.

[FK11] Muhammad Omer Farooq and Thomas Kunz. Operating systems for wireless sensor networks: A survey. *J. of Sensors*, 11(12):5900–5930, 2011.

[GAM+12] Y. Guan, C.A.D. Adi, T. Miyoshi, M. Koibuchi, H. Irie, and T. Yoshinaga. Throttling control for bufferless routing in on-chip networks. In *6th IEEE Intl. Symp. on Embedded Multicore Socs (MCSoC)*, pages 37–44, Sept 2012.

[Geb08] Fayez Gebali. *Analysis of Computer and Communication Networks*. Springer, 1st edition, 2008.

[GI07] L. Gatzoulis and I. Iakovidis. Wearable and portable eHealth systems. *IEEE Eng. in Medicine and Biology Magazine*, 26(5):51–56, October 2007.

[Gib02] Jerry D. Gibson. *The Communications Handbook*. CRC Press LLC, 2nd edition, 2002.

[GKK+03] J.-E. Garcia, A. Kallel, K. Kyamakya, K. Jobmann, J.-C. Cano, and P. Manzoni. A novel DSR-based energy-efficient routing algorithm for mobile ad-hoc networks. In *58th IEEE Intl. Conf. on Vehicular Technology*, volume 5, pages 2849 – 2854, October 2003.

[GKS03] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. Timing-sync protocol for sensor networks. In *1st Intl. Conf. on Embedded networked sensor systems (SenSys)*, pages 138–149, 2003.

[GP10] S.A Gopalan and Jong-Tae Park. Energy-efficient MAC protocols for wireless body area networks: Survey. In *Intl. Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 739–744, Oct 2010.

[GPOL05] Boggia G., Camarda P., Fiume O., and Grieco L.A. CF-MAC and H-MAC protocols for energy saving in wireless ad hoc networks. In *61st IEEE Conf. on Vehicular Technology (VTC)*, volume 4, pages 2560–2564 Vol. 4, May 2005.

[GRA+95a] Zimmerman Thomas G., Smith Joshua R., Paradiso Joseph A., Allport David, and Gershenfeld Neil. Applying electric field sensing to human-computer interfaces. In *Intl. Conf. on Human Factors in Computing Systems (SIGCHI)*, pages 280–287, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[GRA+95b] Zimmerman Thomas G., Smith Joshua R., Paradiso Joseph A., Allport David, and Gershenfeld Neil. Applying electric field sensing to human-computer interfaces. In *Conf. on Human Factors in Computing Systems (SIGCHI)*, CHI '95, pages 280–287, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[GRAMN13] Mohammad Gholami, Hamid Rahimpour, Gholamreza Ardeshir, and Hossein Miar-Naimi. A new fast-lock, low-jitter, and all-digital frequency synthesizer for DVB-T receivers. *Int. J. of Circuit Theory and Applications*, page 13, 2013.

[GRMP07] Ruzzelli A. G., Jurdak R., O'Hare G. M.P, and Van Der Stok P. Energy-efficient multi-hop medical sensor networking. In *1st ACM SIGMOBILE Intel. Workshop on Systems and Networking Support for Healthcare and Assisted Living Environments (HealthNet)*, pages 37–42, New York, NY, USA, 2007. ACM.

[GVLC+13] Sergio Gonza´lez-Valenzuela, Xuedong Liang, Huasong Cao, Min Chen, and Victor C.M. Leung. Body area networks. *Springer J. of Autonomous Sensor Networks*, 13:17–37, 10 2013.

[HCB00] Wendi Rabiner Heinzelman, Anantha Chandrakasan, , and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *33rd Intl. Conf. in System Sciences*, 2000.

[HS06] Jon Hamkins and Marvin K. Simon. *Autonomous Software-Defined Radio Receivers for Deep Space Applications*. Wiley-Interscience, 2006.

[HT06] D. W. Henderson and S. Torn. Verification of the minimum cost forwarding protocol for wireless sensor networks. In *IEEE Conf. on Emerging Technologies and Factory Automation (ETFA)*, pages 194–201, 2006.

[HWZ+12] Ke Huang, Ziqiang Wang, Xuqiang Zheng, Xuan Ma, Kunzhi Yu, Chun Zhang, and Zhihua Wang. A novel clock and data recovery scheme for 10 Gbps source synchronous receiver in 65nm CMOS. In *55th IEEE Midwest Symp. on Circuits and Systems (MWSCAS)*, pages 932–935, August 2012.

[IA14] Sana Ullah Muhammad Imran and Mohammed Alnuem. A hybrid and secure priority-guaranteed MAC protocol for wireless body area network. *Intl. J. of Distributed Sensor Networks*, page 7, February 2014.

[IEE08a] IEEE 1588-2008 standard for a precision clock synchronization protocol for networked measurement and control systems, 2008.

[IEE08b] IEEE. *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. IEEE, July 2008.

[IM05] Mohammad Ilyas and Imad Mahgoub. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. CRC Press LLC, USA, 2005.

[INE13] INESC TEC. The ProLimb project. http://www.inesctec.pt/ctm-en/projects/projects/prolimb/, October 2013.

[JJ01] Jorjeta G. Jetcheva and David B. Johnson. Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks. In *2nd ACM Intl. Symp. on Mobile Ad hoc Networking & Computing*, MobiHoc '01, pages 33–44. ACM, 2001.

[JJB13] S. Jayasimha, P. Jyothendar, and B.Satish Bhargav. Low-complexity DFT-pair carrier acquisition. In *National Conf. on Communications (NCC)*, pages 1–4, February 2013.

[JKJ10] Nelson Bradley J., Kaliakatsos Ioannis K., and Abbott Jake J. Microrobots for minimally invasive medicine. *Annual Review of Biomedical Engineering*, 12(1):55–85, 2010. PMID: 20415589.

[JLH+08] I. Jantunen, H. Laine, P. Huuskonen, D. Trossen, , and V. Ermolov. Smart sensor architecture for mobile-terminal-centric ambient intelligence. *J. of Sensors and Actuators*, 142(1):352–360, 2008.

[JM11] Emil Jovanov and Aleksandar Milenkovic. Body area networks for ubiquitous healthcare applications: Opportunities and challenges. *Springer J. of Medical Systems*, 35(5):1245–1254, October 2011.

[JMOG05] Emil Jovanov, Aleksandar Milenkovic, Chris Otto, and Piet C. De Groen. A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation. *J. of NeuroEngineering and rehabilitation*, 2:6, jan 2005.

[KH14] Shao-Ku Kao and Sheng-Hung Hsueh. A fast-corrected all-digital DCC with synchronous input clock. *Int. J. of Circuit Theory and Applications*, October 2014.

[KKK14] M. Kurosu, F. Koshiji, and K. Koshiji. Electromagnetic field analysis of human body communication between wearable and stationary devices including the earth ground. In *Intl. Conf. on Electronics Packaging (ICEP)*, pages 744–747, April 2014.

[KKP09] Dwivedi A. K., Tiwari M. K., and Vyas O. P. Operating systems for tiny networked sensors: A survey. *Intl J. of Recent Trends in Engineering*, 1(2):152–157, 2009.

[KLLJ13] V.V. Kulkarni, Jung Hyup Lee, Xin Liu, and Minkyu Je. A 100Mbps 0.36mW injection locked clock and data recovery circuit for WBAN transceivers. In *IEEE Intl. Conf. on Microwave Workshop Series on RF and Wireless Technologies for Biomedical and Healthcare Applications (IMWS-BIO)*, pages 1–3, December 2013.

[KLPS11] Young-Sang Kim, Seon-Kyoo Lee, Hong-June Park, and Jae-Yoon Sim. A 110 MHz to 1.4 GHz locking 40-phase all-digital DLL. *IEEE J. of Solid-State Circuits*, 46(2):435–444, February 2011.

[Kon05] Peter Konrad. The ABC of EMG:a practical introduction to kinesiological electromyography. *J. Noraxon INC.*, April 2005.

[KSTP12] E. Karapistoli, D.G. Stratogiannis, G.I. Tsiropoulos, and F. Pavlidou. MAC protocols for ultra-wideband ad hoc and sensor networking: A survey. In *4th Intl. Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 834–841, Oct 2012.

[KSW+08] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. T. Klein Haneveld, T. E. V. Parker, O. W. Visser, H. S. Lichte, and S. Valentin. Simulating wireless and mobile networks in OMNeT++: the MiXiM vision. In *1st Intl. Conf. on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, pages 71:1–71:8, ICST, Brussels, Belgium, Belgium, 2008.

[KTDT12] JeongGil Ko, Nicolas Tsiftesa, Adam Dunkels, and Andreas Terzis. Pragmatic low-power interoperability: ContikiMAC vs TinyOS LPL,. In *9th Annual IEEE Communications Society Conf. on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2012.

[KUU10] Kyung Sup Kwak, Sana Ullah, and Niamat Ullah. An overview of IEEE 802.15.6 standard. In *3rd Intl. Symp. on Applied Sciences in Biomedical and Communication Technologies (ISABEL)*, pages 1–6, Nov 2010.

[KW05] Holger Karl and Andreas Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons Ltd, 2005.

[KWS+11] Sunkwon Kim, Jong-Kwan Woo, Woo-Yeol Shin, Gi-Moon Hong, Hyongmin Lee, Hyunjoong Lee, and Suhwan Kim. A low-power referenceless clock and data recovery circuit with clock-edge modulation for biomedical sensor applications. In *Intl. Symp. on Low Power Electronics and Design (ISLPED)*, pages 347–350, August 2011.

[LBM+11] Benoît Latré, Bart Braem, Ingrid Moerman, Chris Blondia, and Piet Demeester. A survey on wireless body area networks. *J. of Wireless Networks*, 17(1):1–18, 2011.

[LC10] S.M. Lasassmeh and J.M. Conrad. Time synchronization in wireless sensor networks: A survey. In *IEEE Conf. on SoutheastCon*, pages 242–245, 2010.

[LD07] A. Lymberis and A. Dittmar. Advanced wearable health systems and applications - research and development efforts in the european union. *IEEE Magazine on Engineering in Medicine and Biology*, 26(3):29–33, 2007.

[LE02] Kang Lee and John Eidson. IEEE-1588 standard for a precision clock synchronization protocol for networked measurement and control systems. In *34th Annual Precise Time and Time Interval(PTTI) Meeting*, pages 98–105, 2002.

[LG06a] A. Lymberis and L. Gatzoulis. Wearable health systems: from smart technologies to real applications. In *28th IEEE Intl. Conf. on Engineering in Medicine and Biology Society (EMBS)*, volume Supplement, pages 6789–6792, September 2006.

[LG06b] A. Lymberis and L. Gatzoulis. Wearable health systems: from smart technologies to real applications. In *28th IEEE Intl. Conf. on Engineering in Medicine and Biology Society (EMBS)*, volume Supplement, pages 6789–6792, Aug 2006.

[Li08] X. Li. *Wireless Ad Hoc and Sensor Networks, Theory and Applications*. Cambridge University Press, USA, 2008.

[LL08] Shao-Hung Lin and Shen-Iuan Liu. Full-rate bang-bang phase/frequency detectors for unilateral continuous-rate CDRs. *IEEE J. of Circuits and Systems II: Express Briefs*, 55(12):1214–1218, December 2008.

[LLR09] Changle Li, Huan-Bang Li, and Kohno R. Performance evaluation of IEEE 802.15.4 for wireless body area network (WBAN). In *IEEE Intl. Conf. on Communications Workshops (ICC)*, pages 1–5, June 2009.

[LPN+10] Hyung Sun Lee, Choong Bum Park, Kyoung Ju Noh, John Sunwoo, Hoon Choi, and Il-Yeon Cho. Wearable personal network based on fabric serial bus using

electrically conductive yarn. *J. of Electronics and Telecommunications Research Institute (ETRI)*, 32(5):713–721, October 2010.

[LR02] Stephanie Lindsey and Cauligi S. Raghavendra. PEGASIS: Power-efficient gathering in sensor information systems. In *IEEE Conf. on Aerospace*, volume 3, pages 1125–1130, 2002.

[LSG02] Sung Ju Lee, William Su, and Mario Gerla. On-demand multicast routing protocol in multihop wireless mobile networks. *J. of Mobile Networks and Applications*, 7(6):441–453, December 2002.

[LSW09] Christoph Lenzen, Philipp Sommer, and Roger Wattenhofer. Optimal clock synchronization in networks. In *7th ACM Conf.on Embedded Networked Sensor Systems (SenSys)*, pages 225–238, 2009.

[LT10] Huaming Li and Jindong Tan. Heartbeat-driven medium-access control for body sensor networks. *IEEE Transactions on Information Technology in Biomedicine*, 14(1):44–51, Jan 2010.

[LW11] Mei Leng and Yik-Chung Wu. Low-complexity maximum-likelihood estimator for clock synchronization of wireless sensor nodes under exponential delays. *IEEE J. of Signal Processing*, 59(10):4860–4870, 2011.

[Mad08] Upamanyu Madhow. *Fundamentals of Digital Communication*. Cambridge University Press, 1st edition, 2008.

[MAL+14] S. Movassaghi, M. Abolhasan, J. Lipman, D. Smith, and A. Jamalipour. Wireless body area networks: A survey. *IEEE J. of Communications Surveys Tutorials*, 16(3):1658–1686, Third 2014.

[Max01] Dallas Semiconductor. *Understanding and Using Cyclic Redundancy Checks with Maxim iButton Products*, March 2001. App. note 27.

[Max14] Maxim. *1-Wire Protocol*, November 2014. Available in http://www.maximintegrated.com/ products/1-wire/flash/overview/index.cfm.

[MD97] Umberto Mengali and Aldo N. D'Andrea. *Synchronization Techniques for Digital Receivers*. Springer, 1997.

[MDMLN03] Daniel Marco, Enrique J. Duarte-Melo, Mingyan Liu, and David L. Neuhoff. On the many-to-one transport capacity of a dense wireless sensor network and the compressibility of its data. In *2nd Intl. Conf. on Information Processing in Sensor Networks (IPSN)*, pages 1–16, Berlin, Heidelberg, 2003.

[MHA+09] Hanson M.A., Powell H.C., Barth A.T., Ringgenberg K., Calhoun B.H., Aylor J.H., and Lach J. Body area sensor networks: Challenges and opportunities. *J. of Computer*, 42(1):58–65, Jan 2009.

[MHT+05] F. Mizuno, T. Hayasaka, K. Tsubota, S. Wada, and T. Yamaguchi. Development of a wearable computer system with a hands-free operation interface for the use of home health caregiver. *J. of Technology and Health Care*, 13(4):293–300, July 2005.

[Mic12] Microsemi. Libero ide v9.1. Available in http://soc.microsemi.com/download/software/libero/libero91rl.aspx, January 2012.

[Mic13] Microsemi. *IGLOO nano*, November 2013. Available in http://www.microsemi.com/document-portal/doc_download/130695-igloo-nano-low-power-flash-fpgas-datasheet.

[Mil91] D.L. Mills. Internet time synchronization: the network time protocol. *IEEE J. of Communications*, 39(10):1482–1493, 1991.

[MiX11] MiXiM. *OMNeT++ modeling framework for mobile and fixed wireless networks*, June 2011.

[MKC09] Yuce M.R., Ho Chee Keong, and Moo Sung Chae. Wideband communication for implantable and wearable systems. *IEEE J. of Microwave Theory and Techniques*, 57(10):2597–2604, Oct 2009.

[MKSL04] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *2nd Intl. Conf. on Embedded networked sensor systems (SenSys)*, pages 39–49, 2004.

[MLCM05] Inhyuk Moon, Myungjoon Lee, Junuk Chu, and Museong Mun. Wearable emg-based hci for electric-powered wheelchair users with motor disabilities. In *IEEE Conf. on Robotics and Automation (ICRA)*, pages 2649–2654, April 2005.

[MM04] C. Ma and M. Ma. Data-centric energy efficient scheduling for densely deployed sensor networks. In *IEEE Intl. Conf. on Communications*, pages 3652–3656, 2004.

[MM09] Thomas Moscibroda and Onur Mutlu. A case for bufferless routing in on-chip networks. *ACM J. of Computer Architecture News (SIGARCH)*, 37(3):196–207, June 2009.

[MMF98] Heinrich Meyr, Marc Moeneclaey, and Stefan A. Fechtel. *Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing*. John Wiley and Sons, Inc, 1998.

[MVP06] Tommaso Melodia, Mehmet C. Vuran, and Dario Pompili. *The state of the art in cross-layer design for wireless sensor networks*. Springer, 1st edition, 2006.

[MW10] Patel M. and Jianfeng Wang. Applications, challenges, and prospective in emerging body area networking technologies. *IEEE J. of Wireless Communications*, 17(1):80–88, February 2010.

[NBA12] Daniela Negru, Cozmin-Toma Buda, and Dorin Avram. Electrical conductivity of woven fabrics coated with carbon black particles. *J. of Fibers and Textiles in Eastern Europe*, 20(1):53–56, 2012.

[NDM12] E. Nemati, M.J. Deen, and T. Mondal. A wireless wearable ecg sensor for long-term applications. *IEEE Communications Magazine*, 50(1):36–43, January 2012.

[NJM04] Zahi Nakad, Mark Jones, and Thomas Martin. Fault tolerant networks for electronic textiles. In *Intl. Conf. on Communication in Computing*, pages 100–106, 2004.

[ODE+06] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with COOJA. In *31st IEEE Conf. on Local Computer Networks*, pages 641–648, November 2006.

[OMN11a] OMNeT++. *Community*, June 2011. http://www.omnetpp.org/.

[OMN11b] OMNeT++. *Manual*, June 2011. http://www.omnetpp.org/doc/omnetpp/Manual.pdf.

[oNIRPI14] Intl. Commission on Non-Ionizing Radiation Protection (ICNIRP). Non-ionizing radiation protection. Available in http://www.icnirp.org/PubEMF.htm, December 2014.

[Org14] World Health Organization. Electromagnetic fields. Available in http://www.who.int/peh-emf/en/, December 2014.

[OSFM14] T. Ogasawara, A.-I. Sasaki, K. Fujii, and H. Morimura. Human body communication based on magnetic coupling. *IEEE Transactions on Antennas and Propagation*, 62(2):804–813, Feb 2014.

[PB08] A. Pantelopoulos and N. Bourbakis. A survey on wearable biosensor systems for health monitoring. In *30th IEEE Intl. Conf. on Engineering in Medicine and Biology Society (EMBS)*, pages 4887–4890, Aug 2008.

[PB10a] A. Pantelopoulos and N. G Bourbakis. A survey on wearable sensor-based systems for health monitoring and prognosis. *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40(1):1–12, January 2010.

[PB10b] A. Pantelopoulos and N.G. Bourbakis. A survey on wearable sensor-based systems for health monitoring and prognosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40(1):1–12, Jan 2010.

[PBBvR06] Stefan Pleisch, Mahesh Balakrishnan, Ken Birman, and Robbert van Renesse. Mistral: Efficient flooding in mobile ad-hoc networks. In *7th ACM Intl. Symp. on Mobile Ad hoc Networking and Computing*, pages 1–12, 2006.

[PCS+08] Sang-Hune Park, Kwang-Hee Choi, Jung-Bum Shin, Jae-Yoon Sim, and Hong-June Park. A single-data-bit blind oversampling data-recovery circuit with an add-drop FIFO for USB 2.0 high-speed interface. *IEEE J. of Circuits and Systems II*, 55(2):156–160, February 2008.

[PDJ09] T. V. Padmavthy, G. Divya, and T. R. Jayashree. Extending network lifetime in wireless sensor networks using modified minimum cost forwarding protocol-MMCFP. In *5th Intl. Conf. on Wireless Communications, Networking and Mobile Computing (WiCOM)*, pages 1–4, 2009.

[PH07] Lilia Paradis and Qi Han. A survey of fault management in wireless sensor networks. *J. of Network and Systems Management*, 15:171–190, march 2007.

[PHC04] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *2nd Intl. Conf. on Embedded Networked Sensor Systems (SenSys)*, pages 95–107, New York, NY, USA, 2004.

[PHC+11] Jae Won Park, Jin Ha Hwang, Won Young Chung, Seung Woo Lee, and Yong Surk Lee. Design time stamp hardware unit supporting IEEE 1588 standard. In *Intl. Conf. on SoC Design (ISOCC)*, pages 345–348, Nov 2011.

[PO97] E. R Post and M. Orth. Smart fabric, or wearable clothing. In *1st Intl. Symp. on Wearable Computers*, pages 167–168. IEEE, October 1997.

[PPB+12] Shyamal Patel, Hyung Park, Paolo Bonato, Leighton Chan, and Mary Rodgers. A review of wearable sensors and systems with application in rehabilitation. *J. of neuroengineering and rehabilitation*, 9(1):21, April 2012.

[PR99] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. In *IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 90–100, 1999.

[PVM12] P.C. Patel, D.A. Vasavada, and H.R. Mankodi. Applications of electrically conductive yarns in technical textiles. In *IEEE Intl. Conf. on Power System Technology (POWERCON)*, pages 1–6, Oct 2012.

[RAPR13] M.A. Rahman, S. Anwar, M.I. Pramanik, and M.F. Rahman. A survey on energy efficient routing techniques in wireless sensor network. In *Advanced Communication Technology (ICACT), 2013 15th International Conference on*, pages 200–205, Jan 2013.

[RCL+12] Serena Antonia Rubortone, Maria Pia De Carolis, Serafina Lacerenza, Iliana Bersani, Federica Occhipinti, and Costantino Romagnoli. Use of a combined SpO2/PtcCO2 sensor in the delivery room. *J. of Sensors*, 12(12):10980–10989, August 2012.

[RCSA05] Madan R., Shuguang Cui, Lall S., and Goldsmith A. Cross-layer design for lifetime maximization in interference-limited wireless sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1964–1975 vol. 3, March 2005.

[RFR+14] Cavallari R., Martelli F., Rosini R., Buratti C., and Verdone R. A survey on wireless body area networks: Technologies and design challenges. *IEEE J. of Communications Surveys Tutorials*, 16(3):1635–1657, Third 2014.

[RLK+09] Ill-Keun Rhee, Jaehan Lee, Jangsub Kim, Erchin Serpedin, and Yik-Chung Wu. Clock synchronization in wireless sensor networks: An overview. *J. of Sensors*, 9(1):56–85, 2009.

[RN10] Prakash Ranganathan and Kendall Nygard. Time synchronization in wireless sensor networks: A survey. *Intl. J. of on Pervasive and Ubiquitous Computing (UbiComp)*, 1(2):92–102, 2010.

[RRR13] G. Prabhakara Reddy, P. Bhaskara Reddy, and Reddy Reddy. Body area networks. *J. of Telematics and Informatics*, 1(1):36–42, 12 2013.

[SAHW90] Donald B. Sullivan, David W. Allan, David A. Howe, and Fred L. Walls. *Characterization of Clocks and Oscillators*. NIST, 1990.

[SBK05] Bharath Sundararaman, Ugo Buy, and Ajay D. Kshemkalyani. Clock synchronization for wireless sensor networks: A survey. *Elsevier J. of Ad Hoc Networks*, 3:281–323, 2005.

[SE06] Mohanty S.P. and Kougianos E. Biosensors: a tutorial review. *IEEE J. of Potentials*, 25(2):35–40, March 2006.

[Ser10] Pier Andrea Serra. *Biosensors*. InTech, 1st edition, 2010.

[SHH05] Anmol Sheth, Carl Hartung, and Richard Han. A decentralized fault diagnosis system for wireless sensor networks. In *IEEE Intl. Conf. on Mobile Adhoc and Sensor Systems*, pages 194–196, 2005.

[SKLF13] M. Seyedi, B. Kibret, D.T.H. Lai, and M. Faulkner. A survey on intrabody communications for body area network applications. *IEEE Transactions on Biomedical Engineering*, 60(8):2067–2079, Aug 2013.

[SMJ13] P. Shachi, R. Mishra, and R.K. Jatoth. Coherent BPSK demodulator using Costas loop and early-late gate synchronizer. In *4th Intl. Conf. on Computing, Communications and Networking Technologies (ICCCNT)*, pages 1–6, July 2013.

[SMSR02] C.A. Santivanez, B. McDonald, I. Stavrakakis, and Ram Ramanathan. On the scalability of ad hoc routing protocols. In *21th IEEE Intl. Conf. on Computer and Communications Societies (INFOCOM)*, volume 3, pages 1688–1697, 2002.

[SRG08] M. Skrifvars, W. Rehnby, and M. Gustafsson. Proceedings of smart textiles technology and design. In *Coating of Textile Fabrics with Conductive Polymers for Smart Textile Applications*, pages 100–103, June 2008.

[Sta07] William Stallings. *Data and Computer Communications*. Prentice Hall, 8th edition, 2007.

[Ste04] Ransom Stephens. *Jitter Analysis: The Dual-Dirac Model, RJ/DJ, and Q-Scale*. Agilent Technologies, Whitepaper, December 2004.

[SW12] K. Shikada and Jianqing Wang. Development of human body communication transceiver based on impulse radio scheme. In *2nd IEEE CPMT Symp. Japan*, pages 1–4, Dec 2012.

[SY04] F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *IEEE J. of Network*, 18(4):45–50, 2004.

[SZ09] Hang Su and Xi Zhang. Battery-dynamics driven TDMA MAC protocols for wireless body-area monitoring networks in healthcare applications. *IEEE J. of Selected Areas in Communications*, 27(4):424–434, May 2009.

[TDFT+10] C. Tachtatzis, F. Di Franco, D.C. Tracey, N.F. Timmons, and J. Morrison. An energy analysis of IEEE 802.15.6 scheduled access modes. In *IEEE GLOBECOM Workshops (GC Wkshps)*, pages 1270–1275, Dec 2010.

[TEHO11] S. Terada, Y. Enomoto, D. Hanawa, and K. Oguchi. Gait authentication using a wearable sensor. In *Defense Science Research Conference And Expo (DSR)*, pages 1–3, Aug 2011.

[Tex10] Texas Instruments. *MSP430, Ultra low power microcontroller*, November 2010.

[Tex11] Texas Instruments. *CC2420,IEEE 802.15.4 compliant RF transceiver*, April 2011.

[T.G96] Zimmerman T.G. Personal area networks: Near-field intrabody communication. *IBM Systems Journal*, 35(3.4):609–617, 1996.

[tHS08] Ming ta Hsieh and G. Sobelman. Architectures for multi-gigabit wire-linked clock and data recovery. *IEEE J. of Circuits and Systems Magazine*, 8(4):45–57, April 2008.

[Tro05] G. Troster. The agenda of wearable healthcare. In *IMIA Yearbook of Medical Informatics*, pages 125–138, 2005.

[TTKS14] Wang Yun Toh, Yen Kheng Tan, Wee Song Koh, and L. Siek. Autonomous wearable sensor nodes with flexible energy harvesting. *IEEE J. of Sensors Journal*, 14(7):2299–2306, July 2014.

[TTT+09] R. Takeda, S. Tadano, M. Todoh, M. Morikawa, M. Nakayasu, and S. Yoshinari. Gait analysis using gravitational acceleration measured by wearable sensors. *J. of Biomechanics*, 42:223–233, February 2009.

[UHB+12a] S. Ullah, H. Higgins, B. Braem, B. Latre, C. Blondia, I. Moerman, S. Saleem, Z. Rahman, and KS. Kwak. A comprehensive survey of wireless body area networks : on PHY, MAC, and network layers solutions. *J. of Medical Systems*, 36(3):1065–1094, June 2012.

[UHB+12b] Sana Ullah, Henry Higgins, Bart Braem, Benoit Latre, Chris Blondia, Ingrid Moerman, Shahnaz Saleem, Ziaur Rahman, and Kyung Sup Kwak. A comprehensive survey of wireless body area networks: On PHY, MAC, and network layers solutions. *Springer J. of Medical System*, 36:1065–1094, June 2012.

[UKU+09] Sana Ullah, Pervez Khan, Niamat Ullah, Shahnaz Saleem, Henry Higgins, and Kyung Sup Kwak. A review of wireless body area networks for medical applications. *Intl. J. of Communications, Network and System Sciences*, 2(8):797–803, 2009.

[USS+09] Sana Ullah, Bin Shen, S.M., Riazul Islam, Pervez Khan, Shahnaz Saleem, and Kyung Sup Kwak. A study of MAC protocols for WBANs. *J. of Sensors*, 10(1):128–145, December 2009.

[WA07a] Eric Wade and H. Asada. Conductive fabric garment for a cable-free body area network. *IEEE J. of Pervasive Computing*, 6(1):52–58, 2007.

[WA07b] Eric Wade and H. Asada. Conductive fabric garment for a cable-free body area network. *IEEE J. of Pervasive Computing*, 6(1):52–58, 2007.

[WCS11] Yik-Chung Wu, Q. Chaudhari, and E. Serpedin. Clock synchronization of wireless sensor networks. *IEEE Magazine on Signal Processing*, 28(1):124–138, 2011.

[Wil95] Stephen G. Wilson. *Digital Modulation and Coding*. Prentice Hall, 1st edition, 1995.

[WLS+06] Jong-Kwan Woo, Hyunjoong Lee, Woo-Yeol Shin, Heesoo Song, Deog-Kyoon Jeong, and Suhwan Kim. A fast-locking CDR circuit with an autonomously reconfigurable charge pump and loop filter. In *IEEE Asian Conf. on Solid-State Circuits (ASSCC)*, pages 411–414, November 2006.

[WP10] Song Wang and Jong-Tae Park. Modeling and analysis of multi-type failures in wireless body area networks with semi-markov model. *IEEE Communications Letters*, 14(1):6–8, January 2010.

[WTZS10] H. Wang, Q. Teng, X. Zhong, and P.F. Sweeney. Using the middle tier to understand cross-tier delay in a multi-tier application. In *IEEE Intl. Symp. on Parallel & Distributed Processing (IPDPS)*, pages 1–9, 2010.

[XZ09] Jiang Xing and Yunru Zhu. A survey on body area network. In *5th Intl. Conf. on Wireless Communications, Networking and Mobile Computing (WiCOM)*, pages 1–4, Sept 2009.

[YCH+06] Rong-Jyi Yang, Kuan-Hua Chao, Sy-Chyuan Hwu, Chuan-Kang Liang, and Shen-Iuan Liu. A 155.52 Mbps–3.125 Gbps continuous-rate clock and data recovery circuit. *IEEE J. of Solid-State Circuits*, 41(6):1380–1390, June 2006.

[YCL06] Rong-Jyi Yang, Kuan-Hua Chao, and Shen-Iuan Liu. A 200 Mbps–2 Gbps continuous-rate clock and data recovery circuit. *IEEE J. of Circuits and Systems I: Regular Papers*, 53(4):842–847, April 2006.

[YCLZ01a] Fan Ye, A. Chen, Songwu Lu, and Lixia Zhang. A scalable solution to minimum cost forwarding in large sensor networks. In *10th Intl. Conf. on Computer Communications and Networks*, pages 304 –309, 2001.

[YCLZ01b] Fan Ye, Alvin Chen, Songwu Lu, and Lixia Zhang. A scalable solution to minimum cost forwarding in large sensor networks. In *10th Intl. Conf. on Computer Communications and Networks*, pages 304–309, 2001.

[YH11] Hoi Jun Yoo and Chris Van Hoof. *Bio Medical CMOS ICs*. Springer, 1st edition, 2011.

[YJY11] Peng Yu, Song Jia, and Peng Xi Yuan. A self detection technique in fault management in WSN. In *Intl. Conf. on Instrumentation and Measurement Technology (I2MTC)*, pages 1–4, 2011.

[YK12] Mehmet R. Yuce and Jamil Y. Khan. *Wireless Body Area Networks*. Taylor & Francis Group, LLC, 1st edition, 2012.

[YLY09] J. Yoo, Seulki Lee, and Hoi-Jun Yoo. A 1.12 pJ/b inductive transceiver with a fault-tolerant network switch for multi-layer wearable body area network applications. *IEEE J. of Solid-State Circuits*, 44(11):2999–3010, November 2009.

[YMM07] Mengjie Yu, H. Mokhtar, and M. Merabti. Fault management in wireless sensor networks. *IEEE J. of Wireless Communications*, 14(6):13–19, 2007.

[YP09] Fujun Ye and Ruifang Pan. A survey of addressing algorithms for wireless sensor networks. In *5th Intl. Conf. on Wireless Communications, Networking and Mobile Computing (WiCOM)*, pages 1–7, 2009.

[YS09] Kamya Yekeh Yazdandoost and Kamran SayrafianPour. Channel model for body area network (BAN). *Report to the IEEE P802.15.*, page 61, 2009.

[YTT09] Wuyi Yue, Yataka Takahashi, and Hideaki Takagi. *Advances in Queueing Theory and Network Applications*. Springer, 1st edition, 2009.

[Yuc10] M. R. Yuce. Implementation of wireless body area networks for healthcare systems. *J. of Sensors & Actuators*, 162(1):116–129, July 2010.

[YWI+08] Wee Soon Yeoh, Jian Kang Wu, Pek I, Yong Yi Han, Xiang Chen, and Waluyo AB. Real-time tracking of flexion angle by using wearable accelerometer sensors. In *5th Intl. Summer School and Symp. on Medical Devices and Biosensors (ISSS-MDBS)*, pages 125–128, June 2008.

[YZW+11] Zhiyang You, Xibin Zhao, Hai Wan, William N. N. Hung, Yuke Wang, and Ming Gu. A novel fault diagnosis mechanism for wireless sensor networks. *J. of Computer Modeling*, 54(1–2):330–343, July 2011.

[ZCP08a] P. Zicari, P. Corsonello, and S. Perri. A high flexible early-late gate bit synchronizer in FPGA-based software defined radios. In *4th European Conf. on Circuits and Systems for Communications (ECCSC2008)*, pages 252–255, 2008.

[ZCP08b] P. Zicari, P. Corsonello, and S. Perri. A high flexible early-late gate bit synchronizer in FPGA-based software defined radios. In *4th European Conf. on Circuits and Systems for Communications (ECCSC)*, pages 252–255, July 2008.

[ZDD+12] Andreina Zambrano, Fardin Derogarian, Ruben Dias, M. J. Abreu, A. Catarino, A. M. Rocha, José Machado da Silva, João C. Ferreira, Vítor M. Grade Tavares, and M. V. Correia. A wearable sensor network for human locomotion data capture. In *9th Intl. Conf. on Wearable Micro and Nano Technologies for Personalized Health (pHealth)*, June 2012.

[ZGY14] Zhemin Zhang, Zhiyang Guo, and Yuanyuan Yang. Bufferless routing in optical Gaussian macrochip interconnect. *IEEE Transactions on Computers*, 63(11):2685–2700, Nov 2014.

[ZY03] Y. Zhong and D. Yuan. Dynamic source routing protocol for wireless ad hoc networks in special scenario using location information. In *ICCT Intl. Conf. on Communication Technology Proceedings*, volume 2, pages 1287–1290, 2003.

[ZZ11] Hansong Zeng and Yi Zhao. Sensing movement: Microsensors for body motion measurement. *J. of Sensors*, 11:638–660, January 2011.